SÓLO

PROGRAMADORES

Revista especializada para usuarios de PC

AÑO 2. Nº 15
1250 PTAS.

ARGENTINA 9'50 $
CHILE 3000 $



Creación de un compilador a medida

Diseño de servidores Web para Internet

Mixer de Sound Blaster

Hipertexto: control de gráficos y macros

Y además:

- Control de periféricos en Unix
- Programación C++
- Acceso directo a vídeo bajo Windows
- Construcción de demos
- El sonido en Linux

TORNEO DE PROGRAMACIÓN DE DEMOS
250.000 Ptas. en efectivo para el ganador

Iluminación en Ray Tracing



TOWER COMMUNICATIONS, S.R.L


DESDE SIEMPRE TODOS HEMOS DESEADO
MIRAR A TRAVÉS DE WINDOWS



CARTA DEL DIRECTOR



EL CAMINO A LAS 3D Y A LA CYBERCULTURA

Es difícil imaginar un mundo totalmente informatizado en el que el teletrabajo, las autopistas de la información y los avances en el sector multimedia cambien por completo a la sociedad tal y como la conocemos ahora. Algunas películas como *Johny Nenmonic* (basada en un cuento corto de William Gibson, el autor también del *Neuromante* y padre de la *cybercultura*) nos muestran un futuro desolador, en el que las comunicaciones y las líneas de alta velocidad han *encerrado* en sus casas a todos los individuos, delante del ordenador. En esta moderna *cárcel* sin guardianes ni barrotes, la ventana al exterior está representada por la realidad virtual y las 3 dimensiones. Los navegantes huyen del mundo que les rodea a través del cable y la pantalla muestra imágenes y paisajes desaparecidos o inaccesibles a las clases más modestas. Así es la imagen del futuro para muchos escritores de ciencia ficción.

Y en toda esta historia, ¿qué tienen que ver las tres dimensiones? ¿en qué medida afectará esta nueva concepción de la forma de vivir a nuestros equipos informáticos? Sería una divagación interesante. La primera pregunta, ¿software o hardware? Mientras los programadores de juegos exprimen el código para obtener sistemas de gráficos 3D en tiempo real, Creative Labs (los creadores de la saga de tarjetas Sound Blaster) y Argonauts (el grupo de proramación que desarrolló los primeros juegos 3D para Super Nintendo, así como las librerías BRender) anuncian a bombo y platillo el lanzamiento de la 3D Blaster, una tarjeta de *render* vía hardware en tiempo real, con texturas y sombras. El precio inicial de esta maravilla ronda las 40.000 ptas por lo que no es de esperar que se divulgue con rapidez. Sin embargo, como ya ocurriera con la guerra Adlib-Sound Blaster, Creative Labs ha demostrado ser una empresa acostumbrada a ganar grandes y largas contiendas, por lo que no sería de extrañar que en el plazo de 2 ó 3 años la 3D Blaster sea el estándar en cuanto a tarjetas 3D. Cualquiera, con dos dedos de frente puede suponer la gran cantidad de posibilidades que se abren cuando cada hogar tenga un ordenador con un sistema 3D en tiempo real.

De cualquier forma, lo mejor será que cada uno saque sus propias conclusiones. Para mí, el futuro se perfila como una versión light de *Johny Nenmonic* si la sociedad sabe adaptarse a lo que se le viene encima, o *hard* de lo contrario. Nosotros, los que programamos, tendremos mucho que ver en el destino final que se de a nuestro trabajo. En SOLO PROGRAMADORES, vigilaremos y denunciaremos cualquier uso inadecuado que se haga de la tecnología.

De momento disfrutad de este número de SP, que en el próximo habrá más sorpresas. Seguimos recibiendo demos para el concurso e infografías. Todavía estáis a tiempo de participar. Un simple *render* puede haceros ganar un interesante premio en metálico. Os dejo por este mes, el próximo tendréis, entre otras cosas, el reportaje correspondiente al Euskal Party. No os lo perdáis.

MARIO DE LUIS



NOVIEMBRE 1995. Número 15
Es una publicación de



COMMUNICATIONS, S.R.L.

Editor
Antonio M. Ferrer Abelló
Director
Mario de Luis García
Redactor Jefe
Carlos Doral Pérez
Coordinador técnico
Eduardo Toribio Pazos

•••••

Directora Comercial
Carmina Ferrer
Director de Producción
Carlos Peropadre

Colaboradores
Fernando de la Villa, Fernando J. Echevarrieta, Juan M. Martín, Luis Martín, José María Peco, José Carlos Remiro, Enrique de Alarcón, Jorge R. Regidor, Agustín Guillén, Juan Ramón Lehmann, Héctor Martínez, Álvaro Silgado, David Aparicio, Ignacio Cea, Enrique Castañón, Daniel Navarro, Pedro Antón.

Maquetación
Fernando García Santamaría
Tratamiento de imagen
Josefa Fernández Martínez
Servicios Informáticos
Digital Dreams Multimedia, S.L.
Ilustraciones
Miguel Alcón
Secretaría de Redacción
Rosa Arroyo
Suscripciones
Erika de la Riva
Redacción, Publicidad y Administración
C/ Marqués de Portugalete, 10
28027 MADRID
Telf.: 741 26 62 / Fax: 320 60 72
Filmación
Filma Dos S.L.
Impresión
G.D.B.
Distribución
MIDESA

Depósito legal: M-26827-1994
ISSN: 1134-4792

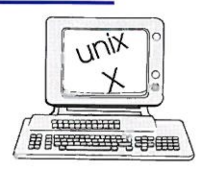
SUMARIO





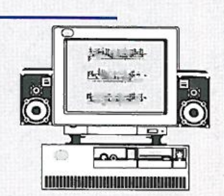


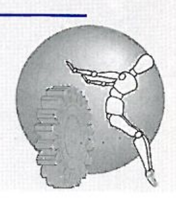


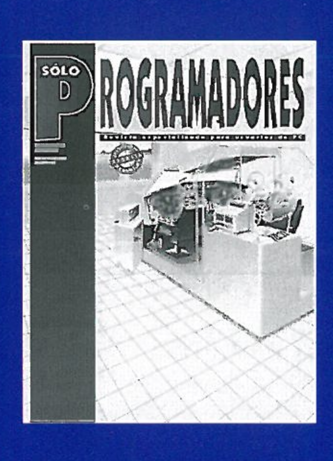












SUMARIO







1er Torneo de Demos e Infografía

El torneo sigue en marcha, en la redacción se arma un revuelo cada vez que llega una nueva demo. Sólo Programadores os anima a concursar. Los que nunca os hayáis presentado a un concurso esta es vuestra oportunidad, y los más veteranos a intentar superarse. Os recordamos que se trata de un campeonato de **Demos** e **Infografía**. Dentro del apartado de demos se incluye también el de **Intros** (como las demos pero más pequeñas). Además del apartado de Infografía se encuentran el de **Vídeo** y el de **Morphing**, de los cuales se habla más adelante.

Existe un suculento premio de **250.000 pesetas en metálico** para los ganadores además de numerosos paquetes de software.

El torneo abrió sus puertas a los más intrépidos el pasado mes de agosto. Durante varios números recogeremos los trabajos realizados por todos aquellos que se animen a participar. El plazo de entrega concluirá el 15 de Diciembre.

BASES DEL CONCURSO

- Para concursar es imprescindible rellenar y enviar la tarjeta de participación que se adjunta en estas páginas.
- El jurado estará compuesto por la redacción de Sólo Programadores.
- Se podrán enviar uno o varios trabajos en cualquiera de las categorías, que son las siguientes:

- DEMOS:

- No entrarán en el concurso aquellas demos que hayan obtenido primeros puestos en campeonatos internacionales. Sin embargo se podrán enviar para incluirlas en el CD-ROM en el que se publiquen los resultados del concurso.
- Existen varias ramificaciones en la categoría de demos:

INTROS

Son demos de longitud mucho más reducida. Habrá dos grupos: de tamaño menor o igual a **4 Kb** (4.096 bytes) y hasta **64 Kb** (65.536 bytes). Tradicionalmente las primeras no suelen incluir sonido, mientras que las segundas suelen aportar alguna melodía o efecto sonoro.

Las longitudes se refieren al tamaño del fichero ejecutable (que vemos desde el DOS). El archivo .EXE o .COM puede estar empaquetado con la utilidad PK-lite. Dicha utilidad comprime un archivo EXE y genera otro EXE de menor tamaño. Este último cuando se ejecuta se descomprime en RAM en tiempo real quedándose con su tamaño original que se supone es superior. Así Intros que ocupan 100 Kb se reducen a 60 Kb con la utilidad PK-Lite o similar, y pueden entrar en la categoría de 64 Kb.

DEMOS

Se entiende por longitud de la demo la suma de las longitudes de los ficheros necesarios para la ejecución de la demo, bien sea un sólo archivo EXE o uno ejecutable más otros de datos. Dichos ficheros también pueden utilizar técnicas de autodescompresión en memoria como las comentadas en el apartado anterior.

Las demos tendrán un tamaño superior a 64 Kb. La única limitación es la de 4 Mb de tamaño máximo de archivos. Se pueden programar demos con extensores de DOS como el DOS4GW de Watcom puesto que el ordenador de prueba será un **486 DX2-66**. Dichos extensores deberán ser enviados en el disquete para poder evaluar la demo.

No se contabilizará como válida la longitud del archivo si está comprimida con uno de los compresores habituales (ARJ, LZH, ZIP, etc.).

- Se enviarán dos copias de la demo. Esto se puede hacer de varias formas. Una consiste en copiar el archivo de la demo dos veces en el disquete (con distinto nombre). Otra forma es enviar dos discos con una copia del archivo en cada uno.

Si es necesario la demo estará comprimida, y si supera la cantidad de 1,4 Mb se podrá empaquetar por volúmenes. **Sólo se admitirán las siguientes compresiones: ARJ, ZIP y LZH.**

En los disquetes irá una pegatina como la que aparece en la **figura 2**. En primer lugar irá el "handle" o nombre del participante/grupo para el concurso y en las casillas se pondrá una X indicando las categorías a las que pertenecen los trabajos enviados en dichos disquetes. Por ejemplo, si un grupo de amigos envía una Intro de 4 Kb en un disquete, pondrá en la pegatina el nombre del grupo de programación y una X en DEMOS, porque la categoría INTROS pertenece a DEMOS.

- INFOGRAFIA:

En las dos categorías, **Imagen** y **Animación**, se admitirán solamente trabajos realizados con programas de infografía o imagen sintética como pueden ser POV, Topas, 3D Studio, etc.).

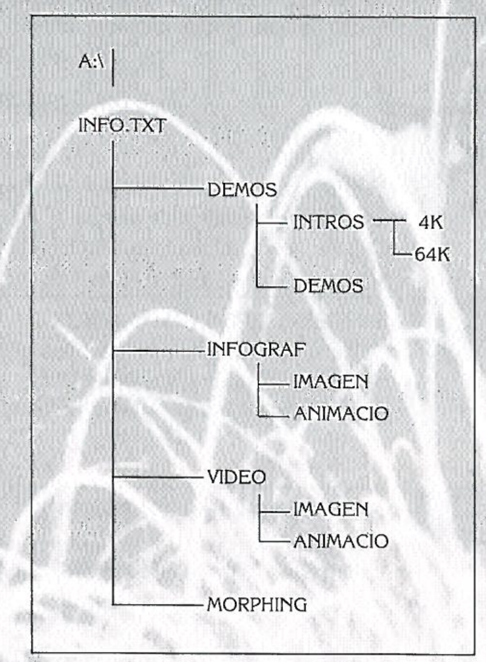


Figura 1.

- **Imagen.** Aquí habrá sub-categorías para englobar los trabajos en función de la resolución y número de colores.
  Sólo serán válidos los trabajos en formato TIF, GIF, JPG y TGA.
  Además de incluir la infografía final se adjuntará obligatoriamente al menos una pantalla del trabajo en fase de construcción. Esto se hará para certificar que el trabajo está siendo realizado por su correspondiente autor.
- **Animación.** En el apartado de infografía en movimiento los trabajos serán admitidos en cualquier formato siempre y cuando se incluya el programa (DOS o Windows solamente) que visualize la animación en un PC 486-DX con 8 Mb RAM y tarjeta True Color VESA.

- VIDEO:

Aquí hay dos grupos similares a la categoría anterior: **Imagen** y **Animación**.

- **Imagen.** Es para los trabajos o composiciones fotográficas en los que se hallan empleado programas específicos para el fotomontaje (Photoshop, Picture Publisher, Photostyler, etc.). Se admitirán sólamente los formatos TIF, GIF, JPG y TGA.

(HANDLE O NOMBRE DE PARTICIPACIÓN)

DEMOS [ ] INFOGRAFIA [ ]

VIDEO [ ] MORPHING [ ]

Figura 2.

- **Animación**. Aquí se engloban los trabajos que hayan sido digitalizados como videos caseros de humor o imágenes con cierto tratamiento digital (efectos de cortina, fades, etc.). Se admitirán solamente trabajos en formato AVI (Video for Windows) y MOV (QuickTime).

- MORPHING:

Se entiende por trabajos de morphing aquellos en los que a través de una secuencia de vídeo se observan transformaciones de objetos, animales o personas. Para estas secuencias no se permite utilizar personajes del ámbito público (políticos, actores, etc.). Sólo se admitirán ficheros AVI, MOV, FLI, FLC y programas que hagan Morphing en tiempo real siempre y cuando hayan sido programados con rutinas propias de los concursantes.

SOBRE EL MATERIAL ENVIADO

Sólo se admitirá el soporte de disquete y cintas de streamer compatibles QIC-80 para enviar los trabajos. Si se desea que se devuelvan los disquetes y las cintas de streamer hay que adjuntar los correspondientes sobres ya franqueados para que Sólo Programadores realice la devolución.

Los disquetes o las cintas donde se envíen los trabajos tendrán una estructura de directorios como la que se muestra en la **figura 1**.

En cada subdirectorio se colocarán el/los programa/s que vayan a participar. En el fichero INFO.TXT se escribirá en formato ASCII el nombre y teléfono de contacto del representante, así como una breve descripción de cada uno de los trabajos que se envíen.

LOS PREMIOS

Se le darán **250.000 pesetas en metálico** a la mejor y más espectacular **Demo o Infografía**. Los finalistas de las otras categorías recibirán numerosos programas de regalo.

TARJETA DE PARTICIPACIÓN

Nombre:..................................Apellidos:..................................Edad: ....

Domicilio: ....Cód.postal: ....Teléfono:..................

Pseudónimo o handle para el concurso: ....

CATEGORÍA DE PARTICIPACIÓN

(En cada modalidad se pondrá el número y los nombres de los trabajos enviados)

- Demos [ ]: ______
- Infografía (ficheros .GIF, .TGA, .TIF, FLI, FLC, etc.)
  Imagen fija [ ]: ______
  Animaciones [ ]: ______
- Vídeo digital (ficheros .GIF, .TGA, .TIF, .AVI, .MOV, etc.)
  Imagen fija [ ]: ______
  Animaciones [ ]: ______
- Morphing [ ]: ______

NORMATIVA PARA EL CONCURSO

- Los trabajos enviados podrán estar realizados por una o varias personas. Si se trata de varias personas solamente el representante aparecerá en la tarjeta de participación. Los datos personales de los otros componentes del grupo serán adjuntados en un folio aparte escritos a máquina.
- No es necesario incluir los fuentes de los programas enviados.
- La responsabilidad de los trabajos enviados atañe tanto al representante que aparece en la tarjeta de participación como a los otros componentes del grupo si es que los hay. Ambos tendrían que responder ante la ley si terceras personas reclamasen la autoría del material o trabajos enviados para este concurso.
- La propiedad intelectual y los derechos de explotación del material enviado pertenecen tanto al representante del grupo como a los demás componentes si los hay. Con la participación en este concurso se concede a la editorial Tower Communications el permiso para publicar en CD-ROM el software enviado para dicho concurso.
- La redacción de Sólo Programadores se reserva el derecho de descalificar a algún participante por un mal funcionamiento del programa enviado o porque dicho programa contenga algún virus informático. También quedarán descalificados aquellos trabajos cuyo contenido sea de mal gusto.
- Enviar esta tarjeta de participación implica aceptar las bases del concurso y la normativa que se estipula en estos apartados.
- La participación en el concurso no será válida si falta algún dato, si hay información incorrecta o bien si la tarjeta de participación es una fotocopia y no la original que viene en la revista. Tampoco será válida la participación en el concurso si la tarjeta enviada no está firmada por el representante.
- La revista Sólo Programadores se reserva el derecho de modificar o ampliar las bases del concurso sobre la marcha si fuese necesario.

Acepto, en representación propia o del grupo, las bases del concurso que aparecen en el número 12 de Sólo Programadores.

Firma del representante:

NOTICIAS



IBM anuncia nuevos servicios Internet para desarrolladores

El Programa de Asistencia a Desarrolladores (DAP) OS/2 de IBM en Europa, Medio Este y África (EMEA) ofrece un servicio de soporte total a desarrolladores a través del World Wide Web de Internet. El cambio a Internet se sucede después de cuatro años de soporte a través de una red de BBS, y ofrece un servicio integrado a más de 5.400 miembros DAP OS/2 en la zona de EMEA.

El DAP OS/2 EMEA de IBM puede encontrarse en las siguientes direcciones World Wide Web (URLs):

- Página Web del DAP OS/2 EMEA de IBM: http://www.europe.ibm.com/getdoc/psmemea/progserv/dap/
- Página Web de IBM Software EMEA: http://www.europe.ibm.com/psmemea/
- Página Web del IBM Solution Developer Operations (SDO): http://www.austin.ibm.com/developer/

Para unirse al DAP OS/2 o para obtener información relacionada se puede enviar un mensaje a la dirección Internet: askibm-rsvpnfo.@info.ibm.com

Para más información:
Información Internet de IBM
Tel: 900 943 533

Novell crea la Personal Access Division

Novell Inc. ha anunciado la organización de la Personal Access Division, una nueva parte de la compañía creada para extender el acceso a red a hogares, colegios y pequeñas empresas. La nueva división es una parte del Novell's Information Access and Management Group (IAMG), dedicado a proveer acceso a la información global.

La Personal Access Division se centrará principalmente en proporcionar paquetes de aplicación de red a los hogares y pequeños negocios con enlaces a servicios en línea, Internet y Servicios de Conexión NetWare de AT&T (ANCS). Novell se ha unido recientemente a America Online para la integración de productos de software con sus servicios comerciales en línea. Por otra parte, Novell ha alcanzado un acuerdo con Netscape para obtener la licencia del navegador World Wide Web denominado Netscape Navigator.

En colaboración con AT&T, Novell eliminará los límites de información y de red con los ANCS, que proporcionará la conectividad de red de área ancha necesaria para uso corporativo, personal y de pequeñas empresas.

Para más información:
Abanico/Hot Line
Liliane Chinyavong
Tel: (91) 594 43 53

Intel crea el foro de implementadores USB

Recientemente ha sido constituido el denominado Universal Serial Bus (USB) Implementers Forum, un grupo abierto a toda la industria cuyo objetivo es el de facilitar el desarrollo de productos con la especificación USB. El foro ha sido fundado por las siete empresas creadoras de la especificación USB: Digital, Compaq, Intel, NEC, IBM PC Company, Microsoft y Northern Telecom.

La adhesión al foro tendrá un coste anual de 2.500 dólares al año y dará derecho a las ventajas no disponibles para los que no son miembros. Algunas de estas ventajas son las siguientes:

- Participación gratuita o a precio reducido a las conferencias de desarrolladores, grupos de tertulia, sesiones de trabajo y eventos de marketing.
- Inserción en el fichero de contactos USB y de catálogos de productos USB.
- Posibilidad de comprar kits de desarrollo.
- Cinco ejemplares gratuitos de cada especificación (distribución automática).

Para más información:
Intel Corporation Iberia
Paseo de la Castellana, 39
28046 Madrid

Tel: (91) 308 25 52
Fax: (91) 310 54 60

Olicom licencia su software ATM a Microsoft

Olicom ha anunciado que su ATM LAN Emulation y software de señales han sido seleccionados por Microsoft para incluirlos en los sistemas operativos Microsoft. Bajo los términos del acuerdo alcanzado por ambas compañías, Microsoft tendrá la licencia del código de señales ATM de Olicom, del software LAN Emulation Client y del software LAN Emulation Server. Este conjunto de software soportará todos los adaptadores ATM actuales y futuros de Olicom, asegurando una total compatibilidad con nuevas versiones de sistemas operativos Windows de Microsoft. Olicom, por su parte, tendrá la responsabilidad de mantener y actualizar el software de acuerdo con las especificaciones del foro ATM.

El ATM LAN Emulation define cómo los sistemas operativos de red Token-Ring y Ethernet, así como las aplicaciones compatibles, pueden funcionar sin cambios en redes ATM. De este modo las aplicaciones existentes pueden utilizar el ancho de banda ATM sin necesidad de un sistema operativo de red ATM nativo.

Para más información:
Olicom Ibérica
Jacobo de Cal
Tel: (91) 345 72 76
Fax: (91) 457 20 79

Intel presenta un nuevo módulo Multibus II

El módulo CH543OSR combina chasis y servicios centrales permitiendo retirar e instalar tarjetas sin necesidad de cortar la corriente (Live Insertion). Este nuevo producto permite a los desarrolladores la creación de sistemas de control que utilicen cualquier procesador y tarjeta de entrada/salida según la especificación Multibus II (IEEE 1296). El campo de aplicación de la tecnología de inserción directa se centra en aplicaciones críticas como instrumentación médica, telecomunicaciones y control de proceso, proporcionando a los desarrolladores la posibilidad de crear aplicaciones robustas, fiables y disponibles en continuo. El chasis CHR543OSR y los módulos de servicios centrales están disponibles al precio de 6.995 dólares por unidad.

Para más información:
Intel Corporation Iberia
Paseo de la Castellana, 39
28046 Madrid
Tel: (91) 308 25 52
Fax: (91) 310 54 60

Nuevo adaptador PCI para ATM 155 Mbps de Olicom

Olicom ha anunciado su primer adaptador PCI para ATM 155 Mbps, diseñado para proporcionar conectividad para estaciones de trabajo y servidores ATM. Se trata de un adaptador de altas prestaciones, de fácil configuración, que proporciona un alto grado de interoperabilidad con productos de otros fabricantes de productos ATM.

El nuevo adaptador viene en dos versiones: conectores estándares de fibra multimodo SONET/SDH y conectores UTP RJ45. Puede ser utilizado para proporcionar una conexión de sobremesa a aplicaciones que precisen de un gran ancho de banda, para servidores en un backbone ATM que soporte sistemas operativos de red estándares, o como plataforma para enlazar LANs clásicas con un backbone ATM utilizando un software de encaminado (router).

Por otro lado, el adaptador permite la construcción de estaciones de trabajo o servidores PCI conectados a ATM basados en una arquitectura estándar PC. Incluye drivers para todos los sistemas operativos de red principales, entre los que se encuentran NetWare, UnixWare, Windows for Workgroups, Windows NT y Windows 95, además de incorporar el software ATM Signaling y LAN Emulation de Olicom. El adaptador ATM PCI (OC-6151 y OC-6152) están disponibles al precio de 995 dólares para la versión de fibra multimodo y de 895 dólares para la versión UTP.

Para más información:
Olicom Ibérica
Jacobo de Cal
Tel: (91) 345 72 76
Fax: (91) 457 20 79

Novell incrementa sus beneficios en un 33%

Novell Inc. ha anunciado unos beneficios en su tercer trimestre fiscal de 102 millones de dólares (0,27 dólares por acción), lo que supone un aumento del 33 por ciento sobre las ganancias ajustadas de 77 millones de dólares en el tercer trimestre fiscal del pasado año. Los ingresos en el trimestre fueron de 538 millones de dólares, alcanzando así un aumento de un 10 por ciento sobre los 489 millones de dólares ingresados en el mismo periodo fiscal de 1994. Durante los primeros nueve meses de 1995 los beneficios aumentaron hasta 279 millones de dólares (0,75 dólares por acción) en comparación con los 0,64 dólares por acción obtenidos en los primeros nueve meses del año pasado. Por su parte, los ingresos en el mismo periodo de 1995 fueron de 1.561 millones de dólares, mientras que en 1994 se registraron 1.432 millones de dólares.

Según Robert J. Frankenberg, presidente y consejero delegado de Novell, el sistema operativo de red NetWare aumentó los ingresos de la compañía hasta el 54 por ciento. Sin embargo, los ingresos en aplicaciones de productividad personal disminuyeron en un 35 por ciento sobre una base anual, mientras que los ingresos crecieron en todas las otras categorías principales.

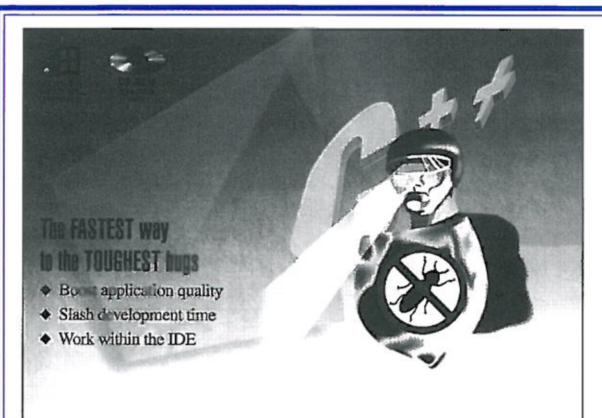
Para más información:
Novell Spain
Daniel Toledano
Tel: (91) 577 49 41

Hitachi anuncia dos nuevas pantallas LCD de bajo consumo

La TX26D68VC1CAA y la TX26D88VC1CAA son dos nuevas pantallas compactas de cristal líquido color TFT (Thin Film Transistor) VGA 640x480 y SVGA 800x600 con un área de visualización de 211,2 mm x 158.4 mm. El consumo de energía de estas pantallas es el típico, incluyendo la retroiluminación de solamente 1,5 y 2,0 vatios respectivamente y pesan 390 gramos. Su encapsulado mecánico mide 244,3 mm x 179,0 mm x 8,0 mm y es compatible con las pantallas S-TN color existentes. Por su parte, los módulos tienen un tiempo de respuesta de 55 ms y pueden visualizar 262.144 colores. Son de retroiluminación CCFL única y tienen un brillo típico de 70 cd/sqm Ambas pantallas incorporan un conversor AC/DC interno y se alimentan con una única fuente de energía de 3,3 voltios.

Para más información:
Hitachi Europe
Pedro Aparicio
Tel: (91) 767 27 82 / 92

NOVEDADES



CodeGuard 1.0 for Borland C++ 4.5

CodeGuard es una utilidad de depuración de código diseñada para la detección automática de errores de programación (bugs) relacionados con la memoria y con recursos RTL. Se integra en el Entorno Integrado de Desarrollo (IDE) de Borland C++ 4.5, permitiendo el salto directo a la línea de código conflictiva. Para habilitar la generación de aplicaciones protegidas por esta utilidad basta con volver a compilar el programa utilizando una nueva librería. CodeGuard permite, entre otras cosas, la detección de problemas en los segmentos de código, datos y montón (heap), así como desbordamientos de memoria, borrados por duplicado e inconsistencias de tipo. Sin embargo, solamente funciona con aplicaciones de 16 bits.

Requerimientos mínimos:

- Windows 3.1, 95 o superior 100% compatible
- Unidad de CD-ROM (disponible en disquettes a través de Borland)
- Borland C++ 4.5
- 9 Mb libres en el disco duro

N O V E D A D E S

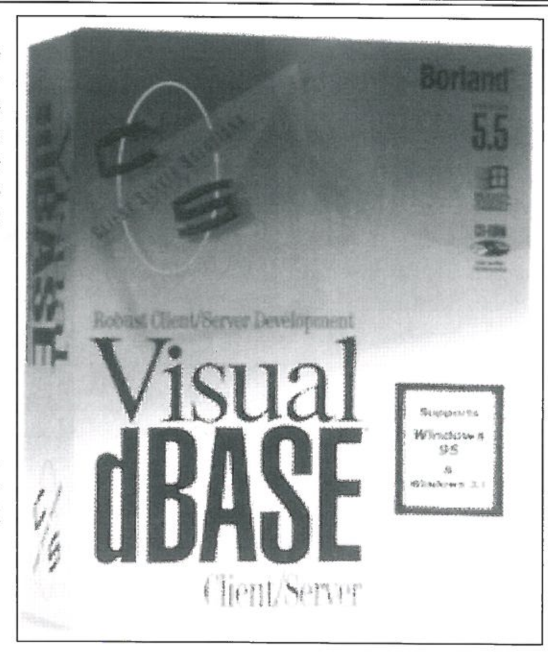
Visual dBASE 5.5

Visual dBASE 5.5 es el último de la familia de productos dBASE. Se trata de una base de datos relacional para Windows que combina las ventajas de las utilidades visuales con el conocido lenguaje dBASE. Incorpora un entorno de desarrollo desde el que se pueden realizar todas las operaciones necesarias para crear y mantener una aplicación de base de datos. Esta versión incluye utilidades (Experts) para ayudar en la creación de tablas, formularios, informes y etiquetas, además de un navegador (Navigator) para moverse a través de los datos.

Otras características de Visual dBASE 5.5 son las siguientes:

- Hasta dos veces más rápido que dBASE para Windows.
- Diseño visual y herencia de formularios y/o controles.
- Desarrollo orientado al objeto conducido por eventos.
- Soporte de OLE, OLE Automation, DDE, VBXs y ODBC.
- Ejecución de aplicaciones dBASE para DOS sin cambios.
- Soporte de características de Windows 95.

El aprendizaje de Visual dBASE es muy rápido, especialmente para los conocedores del lenguaje dBASE. Los usuarios familiarizados con Delphi encontraran muy sencillo el manejo de esta nueva versión de dBASE, puesto que el desarrollo de aplicaciones con ambos productos se realiza de forma similar.



Requerimientos mínimos:

- PC 386 o superior
- Windows 3.1, 95 o superior 100% compatible
- 6 Mb de RAM (8 Mb recomendados)
- 10 Mb libres en el disco duro
- Tarjeta gráfica y monitor VGA/SVGA

LIBROS

The Art and Science of C

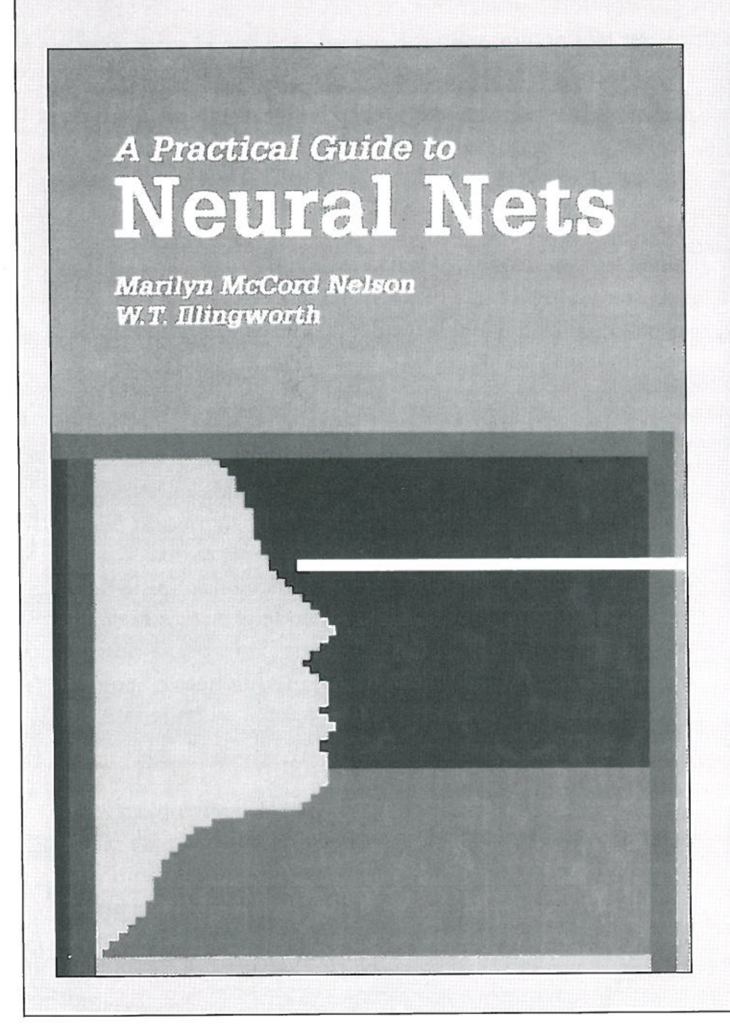
Lejos de ser otro libro más sobre el lenguaje C, The Art and Science of C es una obra que sorprende por su calidad. Se trata de un texto dirigido preferentemente a estudiantes cuyo objetivo principal es el de enseñar los fundamentos de uno de los lenguajes de programación más extendidos del momento. El autor ha enfocado la obra desde el punto de vista de la ingeniería de software, haciendo especial hincapié en el buen estilo de programación. El libro comienza con una introducción a la informática, al software y a la ingeniería de software, para entrar de lleno en la exposición del lenguaje C. Está repleto de ejemplos prácticos, consejos y ejercicios de repaso. Todos los programas incluidos en sus páginas están disponibles vía FTP.

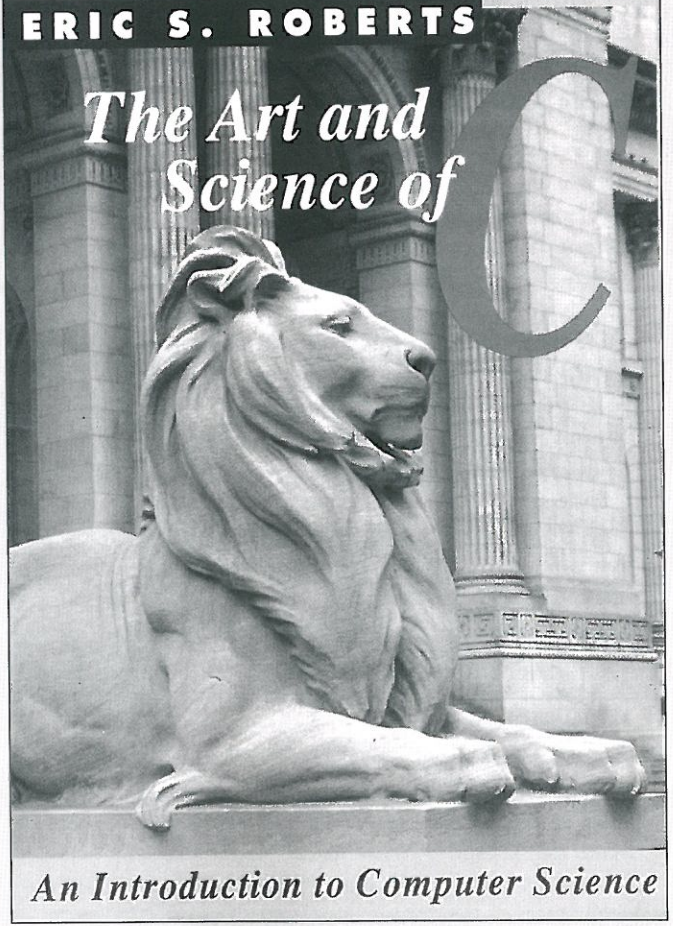
Editorial: Addison-Wesley
Autor: Eric S. Roberts
732 páginas
Idioma: Inglés
Precio: 4.995 Ptas.

A Practical Guide to Neural Nets

Las redes neuronales no son más que una nueva técnica de proceso de información basada en modelos biológicos. Están íntimamente relacionadas con la inteligencia artificial y están alcanzando una gran importancia en la actualidad. Esta obra constituye una breve pero completa introducción a las redes neuronales y a sus campos de aplicación. En cada uno de sus capítulos se da respuesta a una pregunta sobre redes neuronales, exponiendo diversos conceptos como el funcionamiento, la implementación, los usos y el futuro de las redes neuronales. Al final del libro se incluye la simulación del funcionamiento de un elemento de cálculo (neurona) en una hoja de cálculo.

Editorial: Addison-Wesley
Autores: Marilyn McCord Nelson y W. T. Illingworth
349 páginas
Idioma: Inglés
Precio: 4.995 Ptas.





EL ERROR

Parecía planificado, orquestado como una revolución silenciosa de las que estallan a los acordes de una canción convenida en las emisoras de radio. Hace siete días, múltiples copias de mi programa empezaron a fallar en varias oficinas del banco: la alarma se dibujó casi simultáneamente en rostros desconocidos de Santa Cruz de Tenerife, de Gernika, de Ribadeo. Al día siguiente, cayeron Aranjuez, Torrevieja, Portugalete y Sabadell. Mi jefe empezó a ponerse nervioso. "Defcon 2", pensé para mí. Para el Departamento de Defensa de los Estados Unidos, Defcon 2 es el máximo nivel de alerta exceptuando el estado de guerra. Para mí, Defcon 2 significa que ya me puedo ir olvidando de salir a las cinco por una temporada.

Llevo siete días depurando mi programa. Llevo siete noches durmiendo con un ojo cerrado y el otro abierto. Llevo siete jornadas enfrentándome a mi propia creación, intentando sin éxito reproducir en el laboratorio las condiciones del fallo para provocarlo, para que el prisionero acceda por fin a hablar con todos los ficheros de log puestos al máximo de detalle. Llevo siete tardes de lluvia intimando con mi enemigo, uniéndome con él como se une el cazador con la presa, pero también como se une la madre al hijo, porque cada línea de este maldito programa es sangre de mi sangre.

Llevo siete días depurando, y de pronto, ahora que son siete noches de dormir mal y ejecutar mi programa paso a paso de modo interminable como un verdugo reticente, y una lluvia de circunstancias empieza a golpear los cristales, tengo una iluminación, creo que me doy cuenta de que todo esto no es trivial, que el doctor Frankenstein comparte mi dolor cuando se ve obligado a enfrentarse a su monstruo, aunque Saturno, desde su tumba en los cielos, me reproche que bien debí haber devorado a mis hijos antes de sufrir tal destino. Y me rebelo, y pienso que, después de todo, depurar es un verbo estalinista, un verbo que sabe a Gulag, a lavados de cerebro, a estadios convertidos en campos de concentración y atestados de futuros desaparecidos. Yo no soy quién para depurar a una criatura nacida de mí mismo. Si, como dice Kahlil Gibran, nuestros hijos no son nuestros hijos, son una flecha que ya surca el aire y nosotros sólo la mano del arquero, ¿cómo puedo haberme desnaturalizado de esta manera? ¿Cómo puedo traicionar mi naturaleza por un maldito sueldo? ¿Cómo puedo asesinar por dinero?

Ese programa contiene un error. Ese programa lo escribí yo en noches de zozobra, y ese error, esté donde esté, también lo escribí yo, también es mío como son míos mis poemas buenos y malos de la adolescencia, y son míos los besos afortunados y desafortunados que repartí por el mundo. Ahora ese error se ha extendido por la red de oficinas y sus llantos de niño ya se oyen por las calles del Albaicín en Granada, resuenan en las murallas de Ávila, se imponen al ruido de la Gran Vía. Un hombre y una mujer se conocieron en la cola que provocó mi error en la ventanilla. Quizá en este momento, sin saberlo, estén engendrando su primer hijo, y parte de su semilla la germiné yo en un descuido.

Mi jefe quiere que yo haya encontrado el error ayer. Sus amenazas se oyen venir por los pasillos y se suspenden sobre mí como una espada de Damocles. Mi jefe cree, como todos los jefes, que encontrar un error no sólo no es algo doloroso, sino que debería ser fácil. Que existe una manivela que yo giro y el error se aniquila, un botón que yo pulso y el error se evapora. Quiere, como todos los jefes, que yo sea responsable, que yo responda, que yo exhiba las mismas salidas cuando se me alimenta con las mismas entradas, como un programa bien escrito, como un perfecto e inhumano programa exento de errores.

Pero lo que no sabe mi jefe es que han pasado ya siete días de caza y el cazador está cansado. Ya no quiero encontrar el error, ya no deseo salvar mi pellejo ahogando la pequeña revolución nacida de mis dedos. Por el día hago progresos, encuentro interacciones, aíslo los segmentos de código donde se esconde el error que convierte a mi programa en algo humano. Pero por la noche deshago mi trabajo y saboteo mis ficheros, como Penélope deshacía su tapiz para posponer su boda fatal e inevitable.

No sé cuánto aguantaré así. Probablemente un día me echarán de este lugar y contratarán a un cachorro de programador sin escrúpulos que aplastará mi error como un ciempiés inoportuno. Pero no seré yo quien se lamente. Al menos, habré conocido un día de grandeza.

TECNOLOGÍA DEL WWW

Fernando J. Echevarrieta



El WWW es mucho más que un sistema hipertexto: protocolos, clientes, servidores, lenguajes, interfaces, organizaciones y ¿cómo no? dinero. Una mirada seria y directa a la tecnología de información que está revolucionando las inforpistas.

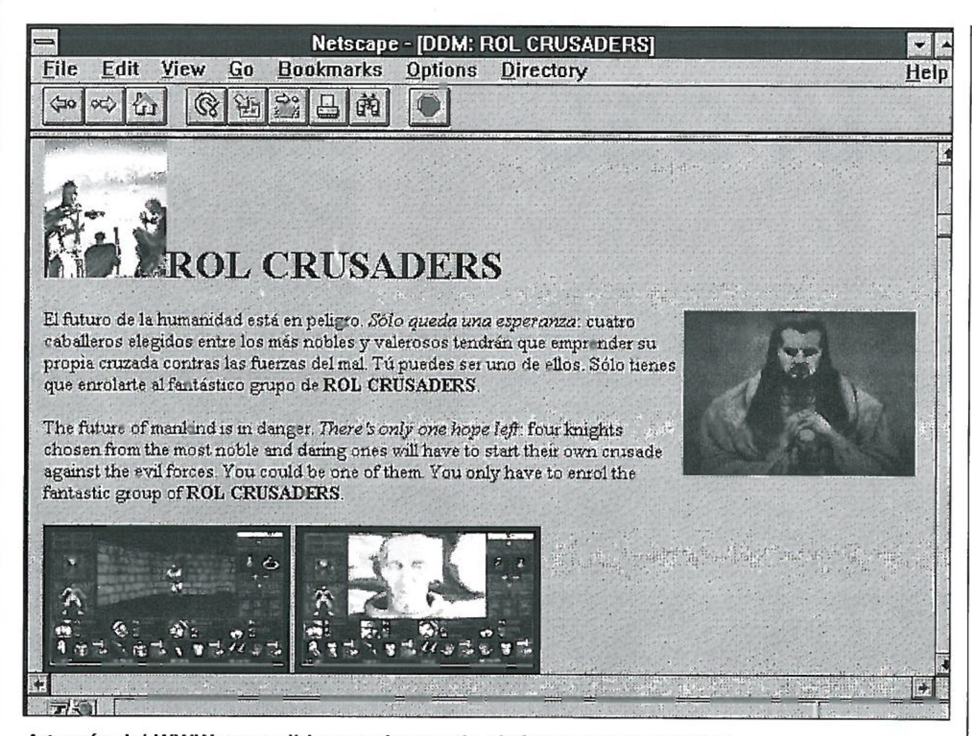
Pues era "casi" verdad. Internet ha llegado. Y ha llegado de la mano del WWW. Ambas son las palabras de moda en todos los medios aunque en pocos se va más alla de la punta del iceberg para mostrar lo que hay detrás: sus posibilidades, su evolución, su tecnología. Y es que el World Wide Web es mucho más que un sistema hipertexto. Con el presente comienza una breve serie de artículos en los que se desvelarán todos los secretos de esta tecnología. En el primero se presentará una panorámica general y los modos de acceder al sistema. Pero ya este primer artículo irá más allá explicando, de forma clara y descriptiva para el público general, los fundamentos de los protocolos que sustentan el sistema. En toda la serie, se indicará la bibliografía adecuada o las especificaciones técnicas apropiadas, la mayoría en forma de *RFCs* (descritos más adelante) o documentos hipertexto que, siempre que se encuentren accesibles en formato electrónico, se facilitarán al lector en los discos o CD-ROMs de la revista.

En la serie se mencionarán todas las propuestas de extensiones al sistema promovidas por empresas e instituciones, así como las alternativas y el estado del arte en general. Se describirá con detalle el *lenguaje de hipertexto HTML*, haciendo especial hincapié en la realización de *forms*, que servirán como introducción de nuevos temas, como la realización de *programas CGI*. A través de estos programas, es posible realizar aplicaciones front-end de acceso a bases de datos previamente existentes y externas al sistema, por lo que en la actualidad, en España existe una gran demanda en el mercado laboral. Debido a su importancia, en este terreno, se profundizará con todo detalle. Se realizarán aplicaciones en shell UNIX y en C, también sobre UNIX, que además servirán como complemento al curso sobre este sistema operativo. Se demostrará la potencia del interfaz CGI integrando con facilidad una aplicacion cliente-servidor con el sistema WWW. Aunque no será necesario conocer cómo ha sido desarrollada esta aplicación para integrarla con el Web, su desarrollo, mediante sockets, será también presentado en el curso sobre UNIX y se tratará de que exista coincidencia en los números de la revista.

Por último, para aquellos lectores que deseen *convertirse en proveedores* del sistema, o para los que simplemente deseen completar sus conocimientos, se les facilitará el software y toda la información necesaria. Pero será mejor comenzar cuanto antes este viaje realizando una primera parada en la

INTRODUCCIÓN, DEFINICIÓN E HISTORIA

Imagine una gigantesca araña lanzando un hilo de un extremo a otro de la Tierra. Imagínela saltando de un país a otro, cruzando océanos y continentes, avanzando y volviendo atrás y en cada salto dejando un hilo. Imagine que cada hilo es un enlace de una fuente de información a otra. La gigantesca tela de araña que cubre todo el plane-



A través del WWW es posible acceder a todo el shareware de internet.

ta es entonces el sistema conocido como World Wide Web (literalmente: "Tela de Araña de Cobertura Mundial").

El World Wide Web (WWW o W3) es hasta ahora el sistema de mayor éxito para el acceso y la distribución de información en Internet. El proyecto surgió en el CERN (European Center for Nuclear Research) como un intento de crear un sistema hipermedia. Según las estadísticas de tráfico en bytes de la National Science Foundation's North American Network (NSFNet), entre los meses de Enero y Agosto de 1993 el tráfico de Internet atribuído al WWW se habia multiplicado por 414 y el uso del servidor del CERN se multiplicaba por dos cada mes. El tráfico de WWW sobrepasó al del Gopher a mediados de 1994 según mediciones sobre el backbone de Internet. Considerando únicamente el flujo de bytes, a finales de 1994, WWW figuraba como el octavo en el ranking entre todos los servicios de red de Internet.

El éxito de esta expansión se debe a que el WWW supone una interfaz que permite que cualquier persona puedar navegar virtualmente por internet sin necesidad de ningún tipo de conocimientos sobre informática ni protocolos. Así como en un sistema hipertexto algunas palabras del texto aparecen destacadas indicando que si se hace click con el ratón sobre ellas conducirán a un nuevo documento, en el Web, textos e imágenes son enlaces a todo tipo de información multimedia que pueda ser encontrada a lo largo de Internet. De esta forma, cada componente multimedia en un documento puede encontrarse en una máquina distinta de la red de redes y servir, además como enlace a otro item o documento hipermedia. De este modo, un simple click de ratón puede realizar una transferencia intercontinental de ficheros o servir de pasarela con cualquier otro servicio de Internet. Simplemente se necesita disponer de un programa cliente de Web, o *browser*, de los que ya existen numerosas versiones para la mayoría de las plataformas, como MS Windows, Macintosh, Amiga, NextStep, X/Dec-Windows y, en modo texto, DOS, UNIX y VMS. Es su sencillez lo que lo hace especialmente indicado como front-end de algunas aplicaciones ya existentes, por lo que los proveedores de todo tipo de información se apresuran a ofrecer su servicios a través del nuevo sistema.

ECHEVA'S EXAMPLE PAGE FOR SOLO PROGRAMADORES
La foto es un enlace
Esto es otro **enlace**
y otro **más**
En esta página **también**
Y de aquí vuelvo **atrás** como una tela de araña

Una fracción de la tela de araña formada por los hiperenlaces.

¿CÓMO ACCEDER?

Todo aquel que desee utilizar el WWW deberá primeramente disponer de una conexión con internet. En la tabla adjunta dispone de los nombres y teléfonos de los proveedores en España en la actualidad.

Abaforum
(902) 10 22 10
Bit Mailer
(91) 403 15 51
Compuserve
01 33 1 47 14 21 60 (Paris)
Goya
(91) 413 48 56
IBM

(900) 99 31 50
Off Campus
(902) 32 00 32
RAN Internet
(91) 535 06 34
Sarenet
(902) 23 90 76
Servicom
(902) 22 66 22

Una vez se cuenta con la conexión será necesario disponer de un browser. En el caso Windows o Mac, bastará pulsar el icono correspondiente y se realizará una llamada por modem automáticamente a la maquina del proveedor correspondiente tras lo cual aparecerá en pantalla lo que se denomina *home page* o página de partida.

Entre los browsers más conocidos se encuentran Lynx, Cello, Arena, Netscape, Mosaic, WinWeb, MacWeb y HotJava. Lynx, para DOS, UNIX y VMS es especialmente interesante para aquellos usuarios que no sispongan de una conexión IP sino únicamente de la posibilidad de acceder a una máquina conectada a internet en modo terminal ya que es el único cliente en modo texto. Una buena solución para los estudiantes universitarios que puedan acceder por modem a su facultad o escuela pero no dispongan de SLIP o PPP (IP sobre línea serie). Cello fue uno de los primeros browsers pero enseguida paso a la cabeza en popularidad el famoso Mosaic, más rápido y con más posibilidades y, posteriormente, el Netscape, incorporando multitud de mejoras y extensiones al lenguaje de hipertexto. Las nuevas generaciones de Mosaic compiten hoy en día por el mercado con Netscape, pudiéndose encontrar versiones de ambos para Mac, Windows y UNIX X-Window. De ellos se habla más adelante en este mismo artículo. Otros browsers como WinWeb o MacWeb, destacan por su compromiso entre sencillez y eficiencia. Por último, el caso especial de HotJava también será comentado más adelante en este mismo artículo, cuando se hayan expuesto algunos conceptos más.

En el próximo CD-ROM de Sólo Programadores se facilitarán versiones Windows de Netscape, Cello, Mosaic y WinWeb y Linux de Netscape con las extensiones adecuadas, en su caso, para poder utilizarlos en modo local para consulta de páginas WWW almacenadas en disco.

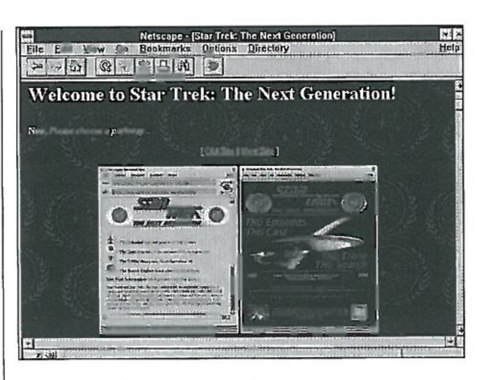
LENGUAJES Y NOMENCLATURA: URL's, HTML, SGML

El proceso de localización de información en Internet se hace cada vez más complejo a medida que crece la red. Por ello, recientemente se está dedicando un gran esfuerzo a la definición de estándares que busquen este objetivo. Así, para localizar los items de información se han definido los llamados *URI*'s o *Universal Resource Identifiers* de los que en el WWW se utilizan los llamados *URL*'s. URL es el acrónimo de *Uniform Resource Locator*, actualmente un estándar en fase de borrador para localización de cualquier entidad, recurso u objeto en internet. El formato utilizado es *método_de_acceso://host.dominio[:puerto]/path/fichero.* donde el *método_de_acceso* es uno de los indicados en la tabla 1 (que aún se encuentra en fase de definición) *host* es la máquina que aloja la información, *dominio* es la "zona" de internet a la que pertenece, y *puerto* es el punto de entrada de los programas de búsqueda que por defecto es el número 80 para servicios de WWW.

file	fichero local o conexión ftp anónima
ftp	conexión ftp anónima
http	conexión http (WWW)
gopher	conexión gopher
news	conexión NNTP a un servidor de News (no poner //)
telnet	sesión telnet
WAIS	conexión WAIS

Está es la forma de indicar a un browser a qué lugar se desea que se conecte. Sin embargo, el usuario, rara vez debe utilizarla ya que todos los browsers disponen de una opción de acceso a guías de internet donde todos los servicios aparecen ya en forma de enlaces hipertexto. Además, también permiten almacenar en una guía particular los lugares preferidos de cada uno.

El WWW utiliza el lenguaje *HTML* (*HyperText Markup Language*) para crear y reconocer documentos hypermedia, relacionando en cada documento, cada enlace con un URL. También otorga un formato al documento, del que dependerá la presentación final que vea el usuario a través de su browser. Un fichero en lenguaje HTML no es más que un fichero ASCII que incluye etiquetas para expresar estos enlaces y/o formatos. Este lenguaje será estudiado con todo detalle el próximo mes (incluyendo la realización de forms y las propuestas para la versión 3.0 o HTML+, en desarrollo)



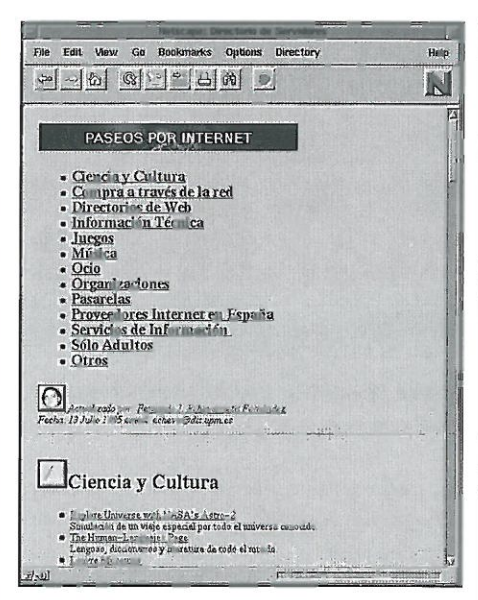
Desde Windows, un browser se maneja igual que la ayuda.

HTML es un lenguaje orientado a sintaxis que forma un subconjunto del *SGML* (*Standard Generalized Markup Language*).

En SGML, el contenido se encuentra separado de la forma o la presentación. Esto quiere decir que un lenguaje SGML puede etiquetar un titular como un "titular" (nivel lógico), pero no llega a especificar como se debe presentar un titular (nivel físico). La ventaja de esto es que de esta forma, se puede configurar en el programa de presentación como se desea ver los titulares. Por contra, diferentes programas con diferentes configuraciones producirán distintas presentaciones de un mismo documento. En cualquier caso, SGML nació con el objeto de convertirse en el estándar universal y definitivo de representación de documentos y precisamente por ello, define estilos lógicos. La idea sería que todo programa fuera capaz de leer y escribir SGML. De ese modo, no sería necesaria la existencia de programas de conversión de cada formato a todos los demás.

¿CÓMO FUNCIONA? O EL PROTOCOLO HTTP

Aunque para el "navegante" Internet, no tiene importancia lo que ocurra



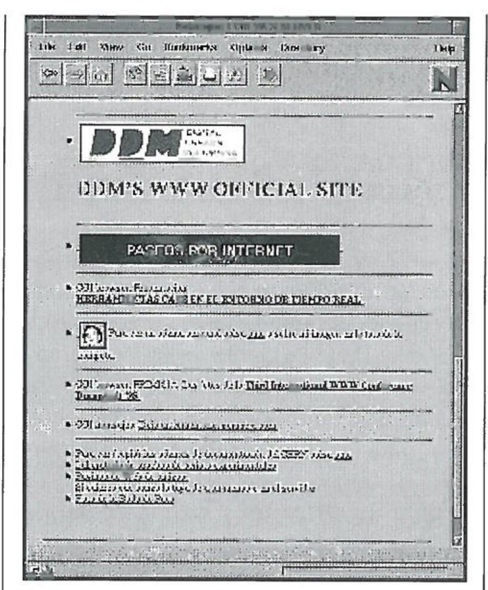
Ejemplo de guía de servicios vista con Netscape para Linux.

"debajo" es interesante conocer cómo funciona internamente el sistema. Estos principios básicos serán, además, fundamentales para aquellos lectores que deseen realizar sus propias aplicaciones CGI.

Para la comunicación entre clientes (browsers) y servidores en el WWW se utiliza el protocolo nativo *HTTP* (*HyperText Transfer Protocol*), todo ello de manera transparente al usuario. Este protocolo se apoya sobre un protocolo de transporte fiable orientado a conexión, generalmente TCP. El cliente establece una conexión con el servidor y envía una petición conteniendo la palabra GET, un espacio y la URL parcial del nodo a ser recuperado terminada con retorno de carro y avance de línea. El servidor responde con el contenido del nodo que consiste en un documento HTML. Así pues, en cada consulta se pueden definir cuatro etapas:

El tráfico en bytes del WWW sobrepasó al de gopher hace más de un año

1. *Conexión*: El cliente llama al servidor. La conexión, si no se especifica lo contrario, se realizará a través del puerto 80 de TCP en la arquitectura de protocolos TCP/IP de Internet.
2. *Solicitud:* El cliente envía la petición



Un servidor WWW puede contener múltiples servicios.

del item de información a recuperar.
3. *Respuesta:* El servidor envía la respuesta.
4. *Cierre:* Se cierra la conexión a ambos lados.

Al usuario de Web, se le crea la ilusión de encontrarse conectado al lugar de donde procede la información; sin embargo, no se trata más que de eso, de una ilusión.

Por poner un ejemplo: si al hacer click sobre un enlace se recupera un documento con dos fotografías, el browser (cliente), realizará una conexión al lugar que indica el enlace y lo solicitará. El servidor remoto le enviará un fichero en formato HTML y después se cerrará la conexión. En ese momento, sin mostrar nada aún, y sin encontrarse conectado a ninguna parte, el browser interpretará el documento encontrando en el código HTML otros dos enlaces a las mencionadas fotografías y realizará una nueva conexión por cada uno de ellos para obtenerlas. Conexiónes que también cerrará en cuanto haya recuperado cada una. De esta forma, se permite que el texto del documento se encuentre en una máquina y cada fotografía en otra distinta. Tras ello, mostrará en pantalla el documento como si se tratara de algo compacto. En este momento, aunque el usuario crea estar conectado con el proveedor de la información, tampoco tendrá abierta ninguna conexión HTTP con ningún servidor.

Esto reporta, además, dos ventajas adicionales: 1. No se están consumiendo recursos para mantener la comunicación ni en el cliente, ni en el servidor, ni se consume ancho de banda de la red y 2. Si el documento recuperado contiene a su vez enlaces hipermedia activables al hacer click con el ratón, se realizará la conexión directamente con el destino, sin que afecte para nada si el provedor del documento original sigue "vivo" o no. De esta forma, es posible también partir de documentos ya almacenados.

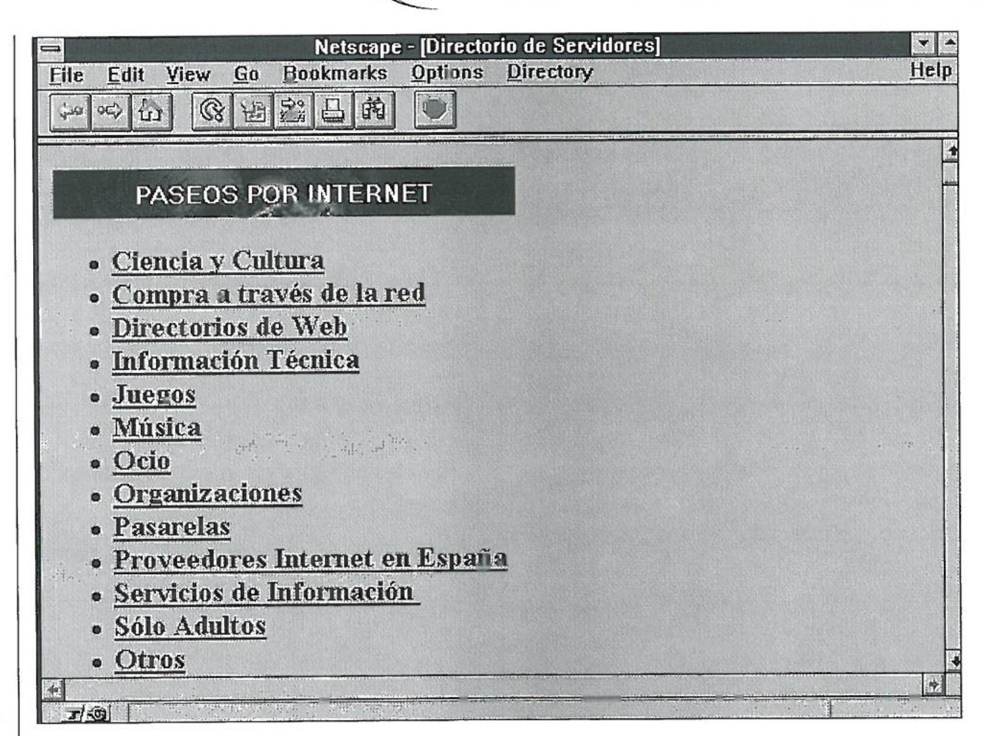
Tras una conexión, el protocolo no guarda memoria de la misma, por lo que cada nueva conexión que se realice será como la primera. Por ello, se dice que HTTP es un protocolo de transmisión sin estado.

Estos son los principios básicos originales del HTTP y del WWW. Sin embargo, los sistemas de información requieren algo más que recuperación, incluyendo front-ends de busqueda y actualización. Para ello, el protocolo define una serie de métodos a los que se hará referencia en el articulo dedicado a CGI. Una última nota a reseñar es que los mensajes transportados por este protocolo son ASCII, en concreto realiza transmisiones de 8 bits ISO Latin 1. Así, es posible realizar una conexión telnet al puerto 80 de una máquina con un servidor WWW, y teclear "hablando en su lenguaje", es decir HTTP.
[Ilustracion]

EVOLUCIÓN EN INTERACTIVIDAD: CGI

Como se ha comentado, el WWW surgió como un mecanismo de comunicación interno, no se esperaba que llegara a convertirse en el interfaz definitivo de Internet (al menos, por el momento). Así pues, la concepción inicial del sistema adolecía de ciertas posibilidades. El sistema se ha extendido vertiginosamente y continuamente surgen nuevas propuestas para incorporar tanto al proto-

colo HTTP como al lenguaje HTML. Realmente, todo evoluciona tan rápido que no sólo estos elementos se encuentran en fase de borrador sino que, como también se ha mencionado, incluso los pilares sobre los que se apoyan (como los URL's) aún se encuentran también en esta fase. De esta forma, todos los estándares lo son "de facto" y surgen versiones y más versiones sin haber aún sido aprobadas oficialmente las anteriores. Así pues, uno de los primeros temas a resolver fue la posibilidad de enviar información a los servidores, tema que se resolvió con los forms incorporados en la versión 2 de HTML propuesta por Dan Connolly. Otro factor importante era la posibilidad de realizar front-ends de otras aplicaciones ya existentes, como el acceso a bases de datos de forma interactiva. Así, el *NCSA* (*National Center for Supercomputing Applications*), creador del archifamoso browser Mosaic, definió en sus servidores un interfaz denominado *CGI* o *Common Gateway Interface*. CGI ha sido adoptado por la totalidad de los fabricantes y proporciona una interfaz entre servidores y programas de aplicación a través de variables de entorno, parámetros y entrada y salida estándar. Mediante el mismo se puede utilizar el WWW como front-end de determinadas aplicaciones a través de un programa intermedio que, por una parte, accede a la aplicación y, por otra, genera dinámicamente documentos HTML.



Un servidor WWW puede contener múltiples servicios.

Se ha propuesto la idea de URL como localizador universal en Internet

NCSA: MOSAIC y CCI

Sin embargo, con CGI no se han resuelto, ni mucho menos, todos los problemas. El asunto parece estar solucionado (al menos en parte) desde el lado del servidor, pero no desde el lado del cliente. Así, debido a que el protocolo HTTP es un protocolo sin estado y cierra las conexiones al enviar un documento, no es posible actualizar dinámicamente la pantalla de un browser. Así, no es posible mostrar un reloj, una animación, las cotizaciones de bolsa o las entradas y salidas de un ciberforo de forma automática. Aunque la información cambie en el servidor, será necesario que alguien en el lado del cliente haga click con su ratón para establecer una nueva conexión que recupere la información actualizada.

Un primer intento de solucionar este problema ha sido también propuesto por NCSA e incluído en las últimas versiones de sus browsers. El *CCI*, *Client Communication Interface*, proporciona un mecanismo análogo de comunicación entre clientes y aplicaciones externas al WWW. De esta forma, se puede controlar el comportamiento del cliente permitiendo que éste actualice la información que muestra en tiempo real o ejecute determinadas acciones sin necesidad de que el usuario haga nada. La idea en principio es buena, el propio programa CGI encargado de acceder y actualizar información podría ser el que le indicara al cliente a través del protocolo CCI cuando debe realizar una nueva petición. En resumen, equivale a reinventar el HTTP haciendo esta vez el cliente de servidor. Sin embargo, mientras en el lado del servidor habrá un programador que estudiará CGI, en el lado del cliente habrá un usuario "de letras", que no tenga obligación de saber nada de protocolos, y el uso de CCI no es transparente ya que se basa en indicar al cliente que "escuche" en un determinado puerto las intrucciones que le enviará otro programa que hará las veces de operador humano. Quizá, por ello, el único fabricante que ha desarrollado el sistema ha sido NCSA y sus licenciatarios en las nuevas generaciones de browsers Mosaic.

NETSCAPE: CLIENT-PULL, SERVER-PUSH, NSAPI

Por su parte, Netscape, que ha tomado la posición de cabeza en el mercado de los browsers, ha resuelto este problema de forma transparente al usuario. Así mediante el mecanismo conocido como *client-pull*, se extiende el lenguaje HTML mediante una nueva instrucción que indica al servidor que debe realizar una nueva conexion finalizado un plazo de tiempo. De esta forma, el servidor no se toca y para mostrar, por ejemplo, una presentación de dispositivas temporizada bastará con que cada documento incluya esta nueva instrucción.

Otra solución que aportan para casos en los que no se pueda conocer de

antemano el momento en que debe realizarse la nueva conversión es el mecanismo *server-push*, basado en la definición de un nuevo MIME type, un tipo especial de datos que evita que el protocolo HTTP cierre las conexiones. De esta forma, las animaciones han comenzado ya a poblar el ciberespacio. Sobre este asunto se volverá en el articulo dedicado a CGI.

Pero los diseñadores de Netscape tampoco están contentos con el interfaz CGI ya que como apuntan, implica la necesidad de ejecutar nuevos programas, que no contemplan la posibilidad del uso compartido de recursos ni de influir directamente en la comunicación WWW. Aunque todo esto es cierto, hay que considerar que las propuestas realizadas por instituciones con ánimo de lucro se encuentran influídas por motivos económicos además de los puramente prácticos y tecnológicos. Sea como fuere, la propuesta de Netscape, llevada a la práctica en su servidor Netscape Server, define un interfaz de servicio o API bautizado como *NSAPI* para acceder a un conjunto de clases de funciones agrupadas según realizan cada paso necesario en una comunicación. Lo interesante del sistema es la posibilidad de sustitutir estas funciones. Así, por ejemplo, sería posible crear una biblioteca para el inicio de recusos compartidos que a su vez fueran heredados por procesos hijos. Asimismo, permite el acceso a las estructuras de datos internas del servidor y realiza una abtracción de los mecanismos de entrada y salida.

EL HOTJAVA DE SUN

Por su parte, SUN ha desarrollado un tipo especial de browser llamado *HotJava* para resolver todos los problemas. El cliente, desarrollado en lenguaje Java, también de SUN (una evolución de C++), define una máquina virtual independiente del hardware que es capaz de ejecutar programas que le envía el servidor WWW o que se le enlazan. De esta forma, es capaz de autoadaptarse a extensiones no estándar de los protocolos que definan distintos fabricantes así como de "aprender" a mostrar cualquier formato presente o futuro de objeto hipermedia. Del mismo modo, dentro del mismo documento de Web, es posible observar la ejecución de programas interactivos. La arquitectura del sistema permite enlazar módulos CCI (o de cualquier otro tipo) en la máquina virtual aunque aún ningún equipo ha comunicado aún haberlos implementado. Hasta ahora, el principal problema es que HotJava únicamente ejecuta sobre Solaris y SUN, como con otros muchos programas, se niega a desarrollar una versión para SunOS en un intento de forzar a todos su usuarios a migrar al nuevo sistema operativo. Esto puede ser un gran error de cara a potenciar el browser e, incluso al futuro de la gigante compañía pero eso... es otra historia.

¿ESTÁ TODO DICHO? HYPERG

HyperG es otro sistema de información hipermedia que puede ser considerado un superconjunto de WWW. Amplia la funcionalidad del mismo facilitando la gestión de los documentos y manteniendo los enlaces fuera de los mismos. De este modo su actualización y transporte queda libre de problemas y permite la generación de hipermedia a partir de documentos sobre los que no es posible realizar modificaciones, por ejemplo, en soporte CD-ROM. Por otra parte, implementa la totalidad de los posibles enlaces hipermedia, que en WWW se limitan a texto e imágenes en origen. HyperG es compatible en modo nativo con Gopher y WWW, en el caso de este último, generando dinámicamente los documentos HTML y estableciendo un diálogo HTTP.

CONCLUSIONES

En el presente artículo se ha expuesto de manera concisa el estado del arte en WWW. Sin embargo, debido a la celeridad con que se está extendiendo esta nueva tecnología, no sería de extrañar que para el momento de la publicación de la revista, ya se estuvieran discutiendo nuevas propuestas. A lo largo de la serie se abrirán paréntesis para mantener informado al lector de los últimos avances. El próximo artículo tratará el lenguaje HTML en profundidad para ser capeces de hacer TODO lo que se puede hacer.

RFC's: BIBLIOGRAFÍA EN INTERNET

Tanto los protocolos de Internet como una gran parte del trabajo que sobre ella se realiza, no pertenecen a ningún fabricante. Por tanto, no es posible conseguir la información de ningún fabricante. Así pues, la información sobre propuestas de protocolos nuevos, variaciones sobre los existentes, definición de estándares y, en general, todo el trabajo que se realiza en Internet, se plasma en unos documentos denominados *RFC*'s (*Request For Comments*) disponibles en formato electrónico de forma gratuíta. Existe una numeración cronológica de los RFC's, por lo que es importante conseguir siempre el último por si anulara alguno anterior. Antes de ser aprobado oficialmente, un RFC se encuentra en fase de *draft* o borrador y no tiene numeración, por lo que se le suele denominar RFCxxx. Los RFC's tomados como bibliografía para este artículo se adjuntan en el disquete de la revista y son los siguientes:

RFC xxx "Hypertext Transfer Protocol", Tim Berners-Lee, CERN.

RFC xxx "Universal Resource Locators", Tim Berners-Lee, CERN. RFC 822 "Standard for ARPA Internet Text Messages". David H. Crocker. Descripcion del formato E.mail

RFC 850 "Standard for Interchange of USENET Messages".

RFC 977 "Network News Transfer Protocol", Kantor & Lamsley.

BIBLIOGRAFÍA

- Computer Networks & ISDN Systems, vol.27, no. 2, Nov.94

Special Issue: Selected Papers of the First WWW Conference

- Computer Networks & ISDN Systems, vol.27, no. 6, Apr.95

Proceedings of the Third International WWW Conference

- Tutorial Notes of the Third International WWW Conference

- IEEE Computer, vol.27, no.10, Oct.94

Ronald J. Vetter, Chris Spell, Charles Ward

Mosaic and the World-Wide Web, pag 49-57

- T. Berners Lee, "The HTTP Protocol as Implemented in W3"

ftp://info.cern.ch:/pub/www/doc/http-spec.txt.Z

- "World-Wide Web Growth", ftp://nic.merit.edu

El autor puede ser encontrado en:

E-mail Internet: echeva@dit.upm.es

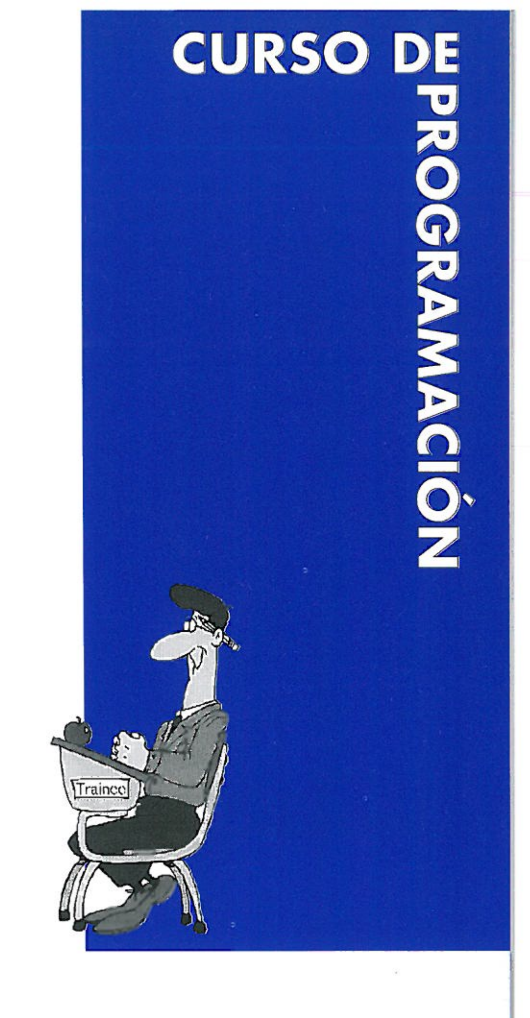
E-mail Compuserve: 100646,2456

WWW: http://highland.dit.upm.es:8000

WWW: http://highland.dit.upm.es:8000/echeva/echeva.html

ORDENACIÓN DE ARRAYS

José C. Remiro



La clasificación y más concretamente la ordenación de un conjunto de elementos y las subsiguientes búsquedas que se realizarán sobre éste son una tarea habitual dentro de la informática, tanto que justifica el esfuerzo realizado por los informáticos en la búsqueda de métodos para realizar esta tarea.

En el presente artículo se describen diversos algoritmos para ordenar los elementos contenidos en un array, también conocidos como métodos de ordenación interna. Estos métodos no son aplicables a la ordenación de archivos, pues a diferencia de éstos en un array todos los elementos se encuentran presentes simultáneamente en la memoria central.

Con el fin de optimizar el uso de la memoria, todos los métodos ordenan los elementos en el propio array. Sin embargo, no todos los algoritmos presentados son óptimos en tiempo. La eficiencia respecto al tiempo de un algoritmo será función del número de intercambios y de comparaciones a realizar entre los elementos a ordenar, cuantos menos intercambios y comparaciones, más eficiente será el algoritmo.

En primer lugar se presentan tres algoritmos básicos denominados directos (inserción, selección e intercambio), indicando cuando corresponda, alguna mejora a realizar. Estos algoritmos son bastante costosos en tiempo (baste decir que el tiempo que consumen es proporcional al cuadrado del número de elementos a ordenar). Tras estos tres algoritmos, se describen el método de Shell que mejora sensiblemente la eficiencia de los anteriores y un método óptimo en tiempo, la ordenación por partición o *quick-sort.* No son los únicos algoritmos de ordenación que existen pero sí los más usuales.

La selección de un método de ordenación estará influenciada por el número de elementos a ordenar. Para un conjunto relativamente pequeño de elementos basta aplicar un método de ordenación básico, pues apenas se notará la diferencia, pero cuando el tamaño del conjunto a ordenar es muy grande está justificada la utilización de un algoritmo óptimo.

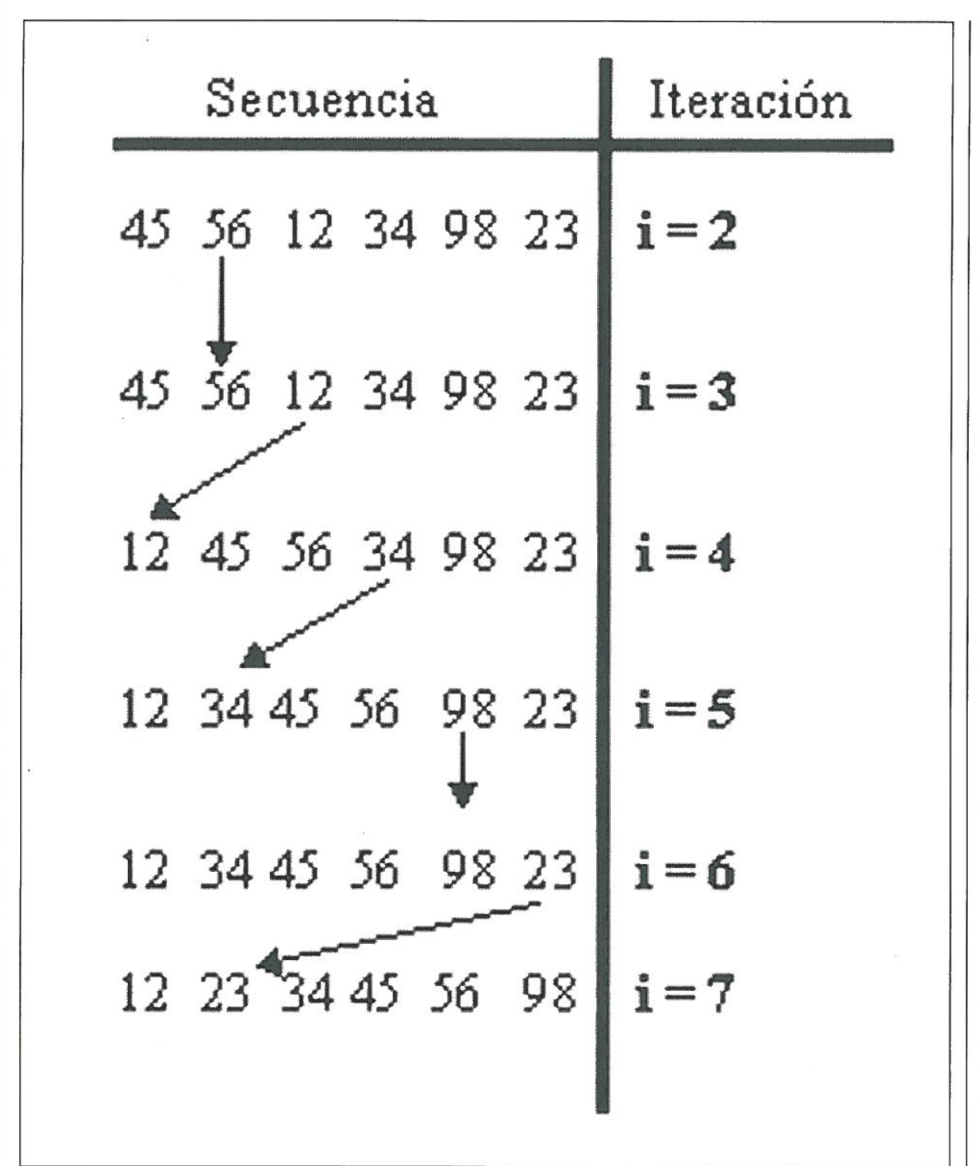
Para ilustrar el funcionamiento de los algoritmos, se ha seleccionado una secuencia de 6 elementos desordenados. Para cada algoritmo se presentará un cuadro en el que se muestra su aplicación a la secuencia.

MÉTODO DE INSERCIÓN

Este método consiste en comparar cada elemento del array con los elementos que le preceden, moviendo éstos una posición hacia delante cuando sean mayores que el elemento que se está intentando insertar en su lugar, de tal forma que al final dejen libre la nueva posición que ocupará el elemento que actualmente se está tratando. El proceso de ordenación para este método se encuentra descrito, utilizando la secuencia de ejemplo, *en el cuadro 1.*

En el cuadro 2 se muestra el algoritmo para el método de ordenación por inserción, se ha utilizado una posición más de las ocupadas por la secuencia de números a ordenar, aprovechando que el método para conocer

La tarea de ordenar un conjunto de elementos es tan frecuente en informática que merece la pena conocer algunos algoritmos para realizarla. La selección de un algoritmo esta influenciada por el tamaño del conjunto a ordenar, además algunas técnicas utilizadas en estos algoritmos también son aplicables a otros procesos.



Cuadro 1.

Pivote para la llamada inicial **ORDEN (1,6)**

45 56 12 34 98 23

Pivote para la llamada **ORDEN (2,6)**

12 56 45 34 98 23

Pivote para la llamada **ORDEN (4,6)**

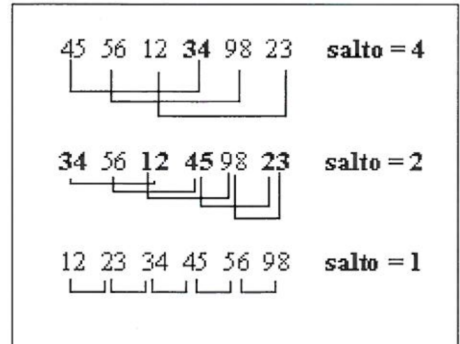
12 23 34 45 98 56

12 23 34 45 56 98 Secuencia ordenada

Cuadro 11.

```
procedimiento net_quick_sort(var a: array (N) real)
 procedimiento orden(izq, dcha: entero)
   sup, inf :entero
   aux, inter: real
  Inicio
   inf= izq
   sup= dcha
   aux= a[(izq+dcha) div 2]
  Repetir
    Mientras a[inf] < aux
       inf= inf + 1
    Fin mientras
    Mientras a[sup] > aux
       sup= sup - 1
    Fin mientras
    Si inf <= sup
      inter= a[inf]
      a[inf]= a[sup]
      a[sup]= inter
      inf= inf + 1
      sup= sup - 1
    Fin si
  hasta inf > sup
  Si izq < sup
   orden(izq, sup)
  Fin si
  Si inf < dcha
   orden(inf, dcha)
  Fin si
 Fin
Inicio /*del proc. ORDEN*/
 orden(1, n)
Fin
```

Cuadro 12.



Cuadro 10.

el punto de inserción está determinado bien al encontrar un elemento menor o igual situado a la izquierda del elemento que está siendo insertado en la actualidad o bien, cuando se ha alcanzado el extremo inferior de la secuencia. La asignación del elemento a insertar dentro del bucle Desde al primer elemento del array evita que el bucle Mientras mas interno tenga una condición más compleja, pues si éste llega a comprobar la posición 1 del array detectará que este elemento es igual y provocará que el bucle termine, insertándose en este caso el elemento en la posición 2 del array, lugar donde realmente empieza la secuencia de ordenación (la técnica de insertar un valor que no pertenece a la secuencia para simplificar las condiciones de terminación se conoce como técnica del centinela).

La mecánica de este algoritmo es bastante sencilla. El bucle más interno Mientras, realiza el desplazamiento y la localización del punto de inserción dado un elemento. El bucle Desde más externo aplica la función de desplazamiento y localización a los elementos comprendidos entre la posición 3 y el último elemento del array. Recuérdese que al final, la secuencia ordenada se encuentra localizada entre las posiciones 2 y el límite superior del array (en este caso N+1).

Se puede mejorar el rendimiento de este algoritmo realizando una búsqueda binaria del punto de inserción, lo que provoca que el número de comparaciones que se realizan sea mucho menor. Este algoritmo se encuentra descrito en el *cuadro 3.* A diferencia del anterior algoritmo, primero se localiza el punto de inserción

```
procedimiento insercion (var a: array (N + 1) real);
 indice, j: entero
 aux: real
inicio
 Desde indice = 3 hasta N + 1
     j = indice - 1
     aux = a[indice]
     a[1] = aux
     Mientras (aux < a[j])
         a[j+1] = a[j]
         j = j - 1
     Fin Mientras
     a[j+1]:= aux
 Fin Desde
Fin
```

Cuadro 2.

```
procedimiento insercion_bin (var a: array (N) real)
 izq, dcha, mit, i, j: entero
 aux: real
inicio
  Desde i = 2 hasta N
    aux = a[i]
    izq = 1
    dcha = i - 1
    Mientras izq <= dcha
        mit = (izq + dcha) div 2
        Si aux < a[mit]
          dcha:= mit - 1
        Si no
          izq:= mit + 1
        Fin si
    Fin mientras
    Desde j = i - 1 hasta izq
      a[j+1] = a[j]
    Fin desde
    a[izq] := aux
  Fin desde
Fin
```

Cuadro 3.

45 56 **12** 34 98 23

12 **56** 45 34 98 **23**

12 23 **45** **34** 98 56

12 23 34 **45** 98 56

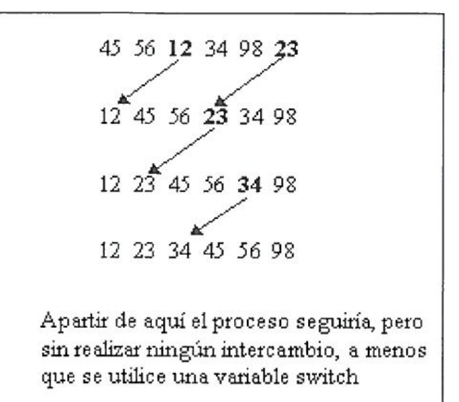
12 23 34 45 **98** **56**

12 23 34 45 56 98

Cuadro 4.

```
procedimiento seleccion (var a: array (N) real)
 i, j, indica_menor: entero
 aux: real
Inicio
 Desde i = 1 hasta N - 1
    indica_menor = i
    aux = a[i]
    Desde j = i + 1 hasta N
       Si a[j] < aux
         indica_menor = j
         aux = a[j]
       Fin si
    Fin desde
    a[indica_menor] = a[i]
    a[i] = aux
  Fin desde
Fin
```

Cuadro 5.



Cuadro 6.

```
procedimiento burbuja (var a: array (N) real)
 i, j: entero
 aux: real
inicio
 Desde i = 2 hasta N
   Desde j = N hasta i
     Si a[j-1] > a[j]
         aux:= a[j-1];
         a[j-1]:= a[j];
         a[j]:= aux
     Fin si
   Fin desde
 Fin desde
Fin
```

Cuadro 7.

y posteriormente se hace avanzar una posición a todos los elementos comprendidos entre el punto de inserción y el lugar donde se encontraba el elemento a insertar.

MÉTODO DE SELECCIÓN

Este método consiste en localizar el menor de todos los elementos y desplazarlo hacia la primera posición del array, a continuación se aplica el mismo proceso con el resto de los elementos, insertando el menor de ellos en la segunda posición, aplicando el mismo mecanismo hasta que solo quede un elemento, en dicho momento el array se encontrará ordenado. El proceso de selección directa se muestra, utilizando la secuencia de ejemplo, en *el cuadro 4.*

El algoritmo mostrado en el *cuadro 5* funciona de la siguiente manera: el índice del bucle Desde más externo indica consecutivamente la posición donde se situará el menor de los elementos, éste se localiza mediante el bucle Desde más interno. La búsqueda del elemento menor se realiza desde la posición indicada por el índice de la instrucción Desde más externa, hasta el extremo superior del array. Una vez encontrado el elemento menor se procede al intercambio entre este elemento y la posición indicada por el índice.

MÉTODO DE INTERCAMBIO

El método de intercambio directo (también conocido como método de la burbuja) se basa en comparar e intercambiar pares adyacentes de elementos, haciendo descender para cada recorrido del array el elemento menor al extremo inferior del array, realizando este proceso con los elementos restantes y finalizando cuando tan solo quede un elemento por ordenar. De los métodos hasta ahora vistos es el menos eficiente. El *cuadro 6* muestra una secuencia del proceso de ordenación.

En el *cuadro 7* se muestra el algoritmo de intercambio directo. Mediante el índice del bucle Desde más externo, se indica la posición donde se insertará el elemento menor (de entre los elementos que se encuentran entre el valor de la variable i y el extremo superior del array). La instrucción Desde más interna se dedica a recorrer el array entre el final del array y el punto donde se insertará el menor de los elementos, comparando elementos consecutivos y "arrastrando" el menor de los elementos hacia la posición indicada por i.

Se puede hacer una mejora de este algoritmo, introduciendo una variable de tipo lógico que realice las funciones de switch, de tal forma que si no se realiza ningún intercambio (sucede cuando todas las condiciones a[j-1] > a[j] son falsas) esta variable permanezca inalterada, posteriormente se comprueba el valor de esta variable, terminando el proceso de ordenación si no se ha producido ningún intercambio.

Se puede realizar otra mejora de este algoritmo si el recorrido por el array se realiza en los dos sentidos opuestos en cada iteración, uno hacia el extremo inferior y otro hacia el extremo superior del array. El *cuadro 8* muestra el algoritmo de intercambio producido por esta mejora, a este nuevo algoritmo se le denomina método de la sacudida.

```
procedimiento sacudida (var a: array (N) real)
 j, k, izq, dcha: entero
 aux: real
Inicio
 izq = 2
 dcha = n
 k = n
 Repetir
   Desde j = dcha hasta izq
     Si a[j - 1] > a[j]
        aux = a[j-1]
        a[j-1] = a[j]
        a[j] = aux
        k = j
     Fin si
    Fin desde
    izq = k + 1
    Desde j = izq hasta dcha
      Si a[j-1] > a[j]
         aux = a[j-1]
          a[j-1] = a[j]
          a[j] = aux
           k= j
       Fin si
    Fin desde
     dcha:= k - 1
 hasta  izq > dcha
Fin
```

Cuadro 8.

```
procedimiento orden_shell (var a: array (N) real)
 aux: real
 paso, i, j: entero
 b: logico
Inicio
 paso= (n + 1) div 2
 Mientras paso > 0
     Repetir
      b= falso
      Desde i= 1 hasta (n - paso)
          Si a[i] > a[i + paso]
             b= verdadero
             aux= a[i]
             a[i]= a[i+paso]
             a[i+paso]= aux
          Fin si
      Fin desde
     hastab = falso
     paso= paso div 2
  Fin mientras
Fin
```

Cuadro 9.

EL MÉTODO DE SHELL

Se trata de un método para mejorar el de inserción directa, tratando de reducir el número de comparaciones y de intercambios.

Este método consiste en seleccionar una secuencia decreciente de "saltos" (se trata de números enteros positivos) terminando en un salto de valor 1. Para cada salto se produce una ordenación de los elementos que distan entre sí el salto correspondiente. Este método está basado en el siguiente hecho: "Si una lista está ordenada de n en n elementos y se ordena de m en m elementos, entonces la lista sigue estando ordenada de n en n".

En el *cuadro 9* se encuentra el algoritmo correspondiente a este método, utilizando un salto inicial de la mitad de los elementos a ordenar y reduciendo el siguiente salto a la mitad del salto anterior, llegando hasta el salto de valor 1. El algoritmo funciona como sigue: la instrucción Mientras más externa se dedica a variar el salto y para cada salto se procede a la ordenación de los elementos que distan el salto correspondiente. La ordenación de los elementos que tienen igual distancia se produce en la instrucción Repetir mas interna. Se dispone de una variable de tipo lógico de tal forma que indica si se ha producido un intercambio, si es así, los intercambios con igual salto realizados con anterioridad se vuelven a comprobar. El proceso de ordenación global finaliza cuando se ha terminado la ordenación de elementos que distan entre sí 1.

El proceso de ordenación para este algoritmo se encuentra descrito, utilizando la secuencia de ejemplo, en el *cuadro 10*.

MÉTODO POR PARTICIÓN

Este método, también conocido como quick-sort, se basa en los principios de intercambio, seleccionándose los elementos a intercambiar en distancias largas.

Quick-Sort se basa en los principios de intercambio

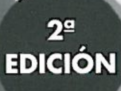
El algoritmo sigue la siguiente mecánica: se selecciona un elemento arbitrario del array, al que se denomina pivote, se busca un elemento situado en posiciones inferiores a la ocupada por el elemento pivote cuyo contenido sea mayor, tras esto se busca un elemento situado en una posición superior a la ocupada por el pivote y cuyo contenido sea menor al del pivote, tras esto se produce un intercambio entre los elementos encontrados. Repitiendo este proceso hasta que el recorrido esté aproximadamente en la mitad del array. En ese momento, se puede afirmar que desde la parte inferior del array hasta la posición ocupada por el pivote todos los elementos son menores que el pivote y que desde el pivote hasta la parte superior del array todos los elementos son mayores que el pivote. Por tanto, bastará con ordenar ambas partes del array por separado para obtener completamente ordenado el array. Pero se puede seguir aplicando este algoritmo por separado a cada una de las partes en que ha quedado dividido el array, puesto que las partes son cada vez más pequeñas se llegará a disponer de partes que consten de un solo elemento, en cuyo caso ya esta ordenada. En el *cuadro 11* se muestra el mecanismo que sigue este algoritmo.

El *cuadro 12* muestra el algoritmo de ordenación por partición recursivo (nótense las llamadas dentro del procedimiento orden a si mismo). En este algoritmo se ha seleccionado como elemento pivote el elemento central de la secuencia a ordenar, que se encuentra delimitada por los parámetros izq e dcha del procedimiento orden. La instrucción Repetir está dedicada al intercambio de los elementos respecto del pivote y las instrucciones Mientras internas a la instrucción Repetir se encargan de localizar elementos adecuados para el intercambio a derecha e izquierda del pivote. Una vez realizados todos los intercambios, si las secuencias tienen más de un elemento (esta condición se corresponde a las condiciones de las dos instrucciones Si) se procede a la ordenación de las partes en que ha sido dividido el array respecto al pivote, mediante las llamadas recursivas orden (izq, sup) (secuencia inferior al pivote) y orden (inf, dcha) (secuencia superior al pivote).

LOS LIBROS DE INFORMÁTICA MÁS ACTUALES AL MEJOR PRECIO

WINDOWS 95







TOWER
Biblioteca de Informática AVANZADA
CONTIENE CD-ROM

Contiene CD-ROM con una selección de ficheros de sonido, vídeo e imágenes, así como las utilidades disponibles actualmente para Windows 95.



EXPLORANDO INTERNET

las Autopistas de la Información



TOWER
Biblioteca de Informática AVANZADA
CONTIENE MAPA Y CD-ROM

Contiene CD-ROM con 800 programas de acceso rápido a INTERNET

3D STUDIO V.4

Las tres dimensiones a su alcance



CADA LIBRO POR SÓLO:

1.995 ptas.
IVA INCLUIDO



Solicite catálogo al Tel. (91) 741 26 62

GESTIÓN PROCESOS

Fernando J. Echevarrieta

A la hora de programar en un sistema operativo serio, llega un momento en que el programador se encuentra con las manos atadas ya que no dispone de permisos para realizar las más sencillas operaciones como puedan ser las de entrada y salida (en adelante E/S). En los casos más delicados, por tratarse de recursos que afectan al funcionamiento del sistema, todos estos servicios se encuentran reservados al sistema operativo y gestionados por sus planificadores internos. Es por ello que para hacer uso de estos recursos, es necesario realizar una serie de system calls (llamadas al sistema), solicitando al S.O. que realice una serie de servicios que no se pueden conseguir de otra forma. Es a la utilización de estas llamadas a lo que se denominará programación de sistema en adelante.

Una system call se presenta al programador como una rutina más de biblioteca en C, aunque durante su ejecución, es el sistema operativo el que toma el control. Así, el conjunto de llamadas al sistema constituye la API del mismo (todos estos conceptos fueron introducidos en el primer capítulo de la serie). En el presente artículo se presentará la parte de la API dedicada a generación y control de procesos. Primeramente se introducirán unas nociones básicas sobre arquitectura de sistemas operativos mientras se describe el sistema de gestión de procesos UNIX. En segundo lugar se describirán las estructuras de datos manejadas por el gestor para pasar a la parte principal en la que se realizará una descripción de la interfaz de llamadas al sistema destinadas a gestión de procesos.

Tras haber estudiado en anteriores artículos el sistema desde el punto de vista del usuario, a partir del presente, el punto de vista se ampliará también al del de programador, en algunos casos y administrador del sistema en otros.

EL GESTOR DE PROCESOS UNIX

Como en todo S.O. concurrente la gestión de procesos es llevada a cabo por el propio kernel. El núcleo se encargará de gestionar el uso de la CPU así como de todos los recursos físicos y lógicos del sistema como se describía en el primer capítulo de la serie. Para ello, todo S.O. debe llevar un seguimiento de todos los datos asociados a cada proceso a través de una estructura de datos denominada tabla de procesos.

Para evitar confusiones, será conveniente distinguir la ideas de proceso, programa y ejecutable. Se designará programa a una secuencia ordenada de instrucciones máquina mientras que ejecutable será la estructura de información (fichero y cabeceras) que contiene el programa y la información necesaria para que el S.O. sea capaz de ejecutarlo. Por último, proceso será el nombre que en adelante se dará a un programa en ejecución.

UNIX planifica el uso de CPU basándose en una estructura de colas multinivel realimentadas basadas en prioridades con una asignación de servicio según un algoritmo de Round-Robin. El significado de esto se explica a continuación. El algoritmo de Round-Robin, es comunmente conocido como de "rodajas de tiempo" ya que divide el tiempo en rodajas y reparte estas rodajas entre los procesos. La planificación de la CPU es pre-emptive, lo que coloquialmente se suele denominar "multitarea real", es decir, que el S.O. puede "quitar" la CPU a un proceso que la ocupa aunque este no desee cederla. El concepto de cola reune a los procesos en espera de recibir la CPU y la idea de

multinivel representa distintas colas según prioridad (algo así como la cola del ciudadano de a pie y la cola de los VIP). El hecho de ser realimentadas significa que cuando termina la rodaja de tiempo de un proceso, éste vuelve a la cola.

Respecto a la asignación de prioridades, UNIX lleva a cabo un ajuste dinámico de las mismas. Así, por una parte, favorece los trabajos interactivos aumentando la prioridad de los procesos que lleven tiempo suspendidos por una operación E/S y disminuye la prioridad de los que acumulan mucho tiempo de uso de CPU. El planificador de procesos incorpora la técnica de pasado reciente, que consiste en la penalización de los procesos que hacen uso intensivo de la CPU en condiciones de fuerte carga.

En cualquier caso, por supuesto, el propio núcleo del S.O. es el proceso más prioritario, pudiéndose realizar una clasificación de las prioridades en NO INTERRUMPIBLE, INTERRUMPIBLE, UMBRAL y DE USUARIO.

TABLA DE PROCESOS

Volviendo a la idea de proceso, así como Wirth describía en su clásica ecuación

ALGORITMOS + ESTRUCTURAS DE DATOS = PROGRAMAS

en el caso de un proceso, como programa en ejecución, hay que incluir un tercer sumando a esta ecuación. Así, un proceso constará de los clásicos componentes: conjunto de instrucciones y estructuras de datos y, además, de un contexto creado por su asignación dinámica en una máquina a cargo de un sistema operativo.

Parte de este contexto, es el descriptor de proceso, manifestación para el S.O. de que existe ese proceso y que se materializa como una entrada en la tabla de procesos. En la figura 1 se puede apreciar una representación de un descriptor de proceso genérico donde se ha reflejado el PID (Process IDentifier), UID (User IDentifier), state (estado), events (eventos a la espera), size (requerimientos de memoria), locations (puntero a u_area, descrita más adelante), priority (paramétros para el gestor de procesos), signals (cola de señales recibidas y aún no procesadas) y accounting (información de tiempos de ejecución, espera, uso del kernel, etc).

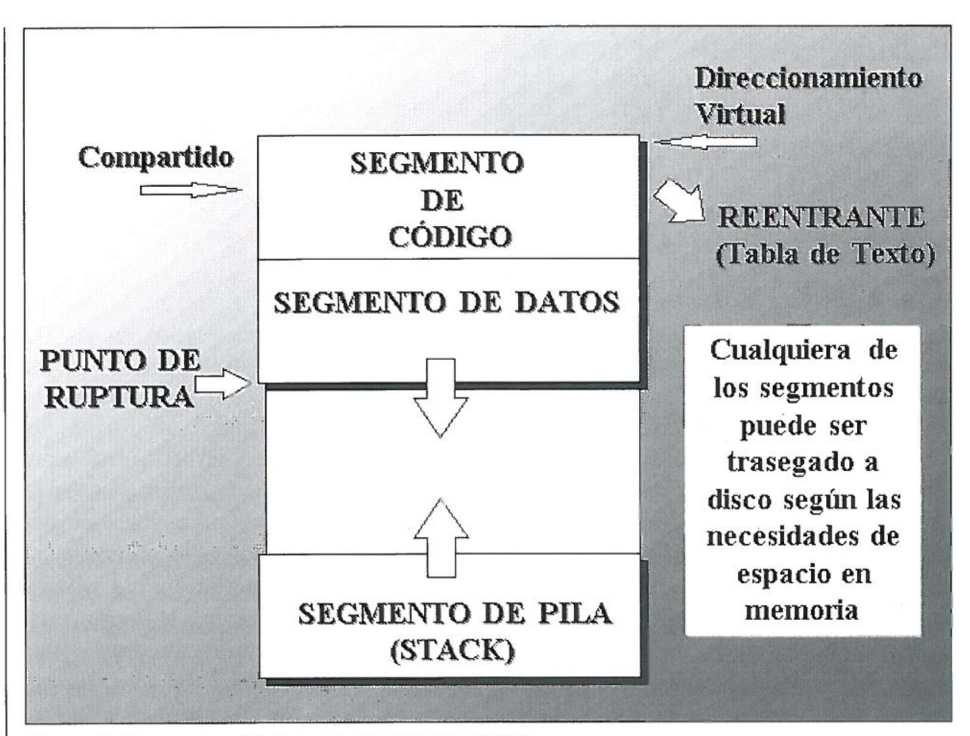


Figura 1. Segmentos lógicos de un proceso UNIX.

Existe otra estructura de datos asociada a cada proceso denominada u_area que proporciona al S.O. información sobre descriptores de fichero abiertos, parámetros, valores de retorno y error de llamadas al sistema, directorio actual y directorio raíz del sistema de ficheros, estructuras de datos de E/S y un puntero a su entrada en la tabla de procesos. Esta zona pertenece al espacio de direccionamiento del kernel, no a la del proceso, por lo que sólo podrá ser modificada por el núcleo.

ESTADOS DE UN PROCESO UNIX

Desde que se inicia un proceso en cualquier sistema operativo, hasta que ter-

Identificador de proceso	PID
Identificador de Usuario	UID
Estado del Proceso	State
Eventos que espera	Events
Memoria requerida	Size
Puntero a u_area	Locations
Parámetros del gestor	Priority
Señales aún no procesadas	Signals
Estadísticas	Accounting

Figura 2. Descriptor de proceso UNIX.

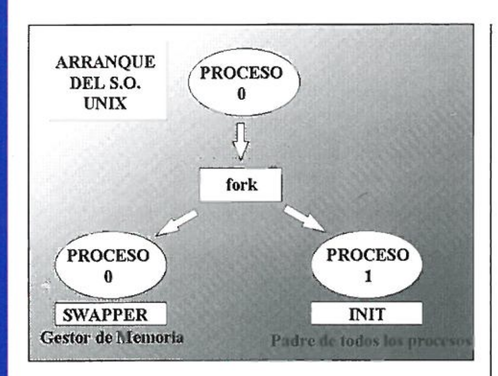


Figura 3. Arranque del S.O.

mina, puede pasar por diferentes estados. En UNIX se distinguen:

1. running o en ejecución

Estado que se podría denominar "normal" en el que el proceso o una rutina llamada por el ocupa la CPU. Así, la ejecución puede llevarse a cabo en modo usuario, cuando es una instrucción máquina del programa de usuario la que ocupa la CPU o en modo kernel o supervisor, en el que se ha producido una interrupción o una llamada al sistema y es el kernel el que ocupa la CPU prestando un servicio al programa de usuario.

2. asleep o suspendido

Estado en el que el proceso se encuentra suspendido, por lo que no ocupa la CPU. Este estado se suele producir cuando el proceso se encuentra a la espera de algún evento, fin de E/S, terminación de otro proceso, etc. En lugar de realizar una espera activa, el planificador del S.O. lo suspende realizando un cambio de contexto hasta que se produzca el evento en cuestión, momento en que despertará y pasara al estado ready.

3. ready o preparado

Un proceso se encuentra preparado cuando se encuentra en cola de espera para recibir la CPU. Es un proceso que se encuentra ejecutándose pero debido a la concurrencia aparente del sistema debe esperar a su turno de CPU.

4. zombie

Cuando un proceso finaliza su ejecución, muere y, aunque el proceso como tal deja de existir, deja los resultados de su "autopsia" en forma de estructura de datos en el kernel con información sobre su estatus de salida, causa de muerte, estadísticas, etc. Se le sigue pudiendo ver pero como es un proceso muerto, ya no se le puede matar, por lo que recibe el nombre de zombie. Será necesario eliminar estas estructuras de datos de otra forma para no llenar de "basura" el sistema (ver wait).

FORMATO DE UN EJECUTABLE

Como se mencionaba en el primer punto, un ejecutable es un programa rodeado de más información necesaria para que el S.O. sea capaz de transformarlo en un proceso, es decir, de ejecutarlo. Así pues, consta de cinco zonas de datos, las dos primeras necesarias y el resto opcionales. A saber:

Una system call se presenta como una rutina más de biblioteca en C

1. Cabecera: contiene información necesaria para la ejecución y los tamaños de las otras secciones en bytes. Contiene un número mágico, un entero largo con información sobre el punto de entrada del programa.
2. Texto y Datos: La zona de texto corresponde al código del programa que será una zona de sólo lectura mientras que la de datos será de lectura/escritura.
3. Información de relocalización.
4. Tabla de Símbolos.
5. Tabla de Strings.

Estas tres últimas partes, opcionales, se suelen emplear para procesos de depuración (por ejemplo, son necesarias con el depurador dbx) y se pueden eliminar mediante el comando strip para reducir el tamaño del ejecutable. También es posible realizar la carga en memoria a través del comando ld -s.

SEGMENTOS LÓGICOS DE UN PROCESO

Al cargar un ejecutable en memoria, el S.O. realiza una división de sus espacio de direccionamiennto en tres segmentos lógicos, como describe la figura 2.

1. El segmento de TEXTO

El segmento de texto comienza el la dirección virtual 0 y aloja el código del programa. Se trata de un segmento protegido de sólo lectura. Esto lleva asociadas una ventaja y un inconveniente. La ventaja consiste en que el código es reentrante, es decir puede ser ejecutado por varios procesos. De esta forma, si dos usuarios arrancan un mismo programa en la misma máquina, se crearán dos procesos que compartirán el mismo segmento de texto, cada uno con un puntero de instrucción distinto. Con ello, el programa sólo se cargará una vez en memoria. El inconveniente viene dado por el hecho de que al tratarse de código compartido y protegido, en UNIX no será posible realizar código automodificable.

2. El segmento de DATOS

El segmento de datos comienza donde termina el de texto y se extiende hasta un punto denominado punto de ruptura que permanece fijo durante la ejecución del proceso. La extensión de los segmentos de datos y texto viene dada por

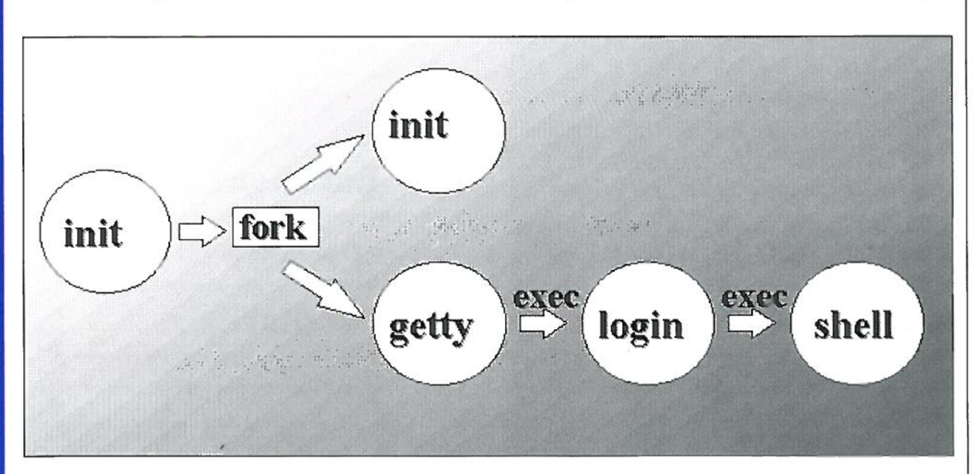


Figura 4. Generación de una shell mediante el paradigma de generación de procesos de UNIX.

el número mágico de la cabecera del ejecutable, que no se carga en memoria.

3. El segmento de PILA (STACK)

El segmento de pila ocupa las direcciones más altas del espacio asociado al proceso y crece hacia las bajas teniendo como límite superior el punto de ruptura. El espacio entre el segmento de pila y el de texto no pertenece al espacio de direccionamiento del proceso. Es posible modificar la situación del punto de ruptura a través de la llamada al sistema brk.

COMANDO PS

El comando ps muestra los procesos en ejecución como es ya conocido al tratarse de uno de los primeros expuestos en esta serie. En esta ocasión se revisita como recordatorio y para ampliarlo con las opciones -a (muestra los procesos "interesantes"), -l (formato largo), -g (procesos no tan interesantes, como getty), -x (incluye demonios, procesos no asociados al terminal), -u (todos los usuarios)

CREACION DE CLONES: fork

A través de la llamada al sistema fork, el kernel realiza una duplicación exacta del proceso llamante, código y datos.

Cada proceso se encuentra representado en la tabla de procesos del sistema

La llamada se define de la forma
int fork();
y devuelve un valor 0 al proceso hijo y el PID del hijo al proceso padre. En caso de no poder realizarse la clonación, el valor de retorno será -1. La forma usual de utilizar la llamada es:

```
int pid;
...
if ((pid = fork()) == 0) {
   /* Codigo del hijo */
}
/* Codigo del padre */
```

Las aplicaciones son innumerables. Desde la más simple, empleada por muchos demonios del sistema que consiste en pasarse automáticamente a background (1. Generación de un clon que se ejecuta en background y 2. Salida), hasta la realización de servidores múltiples concurrentes. En la figura 3 se representa de forma gráfica el mecanismo de arranque del S.O. a través de la reprodución del proceso inicial.

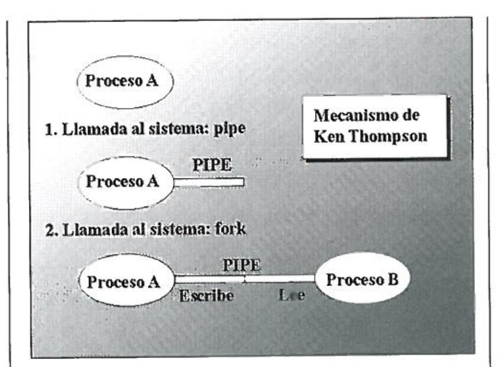


Figura 5. Representación del mecanismo de Ken Thompson.

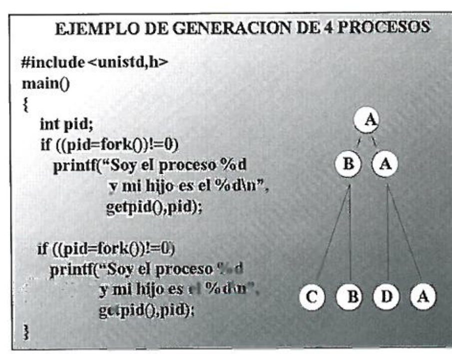


Figura 6. Un sencillo ejemplo sobre la utilización de fork.

Desde el punto de vista conceptual, un fork produce, como se ha expuesto, una duplicación total del proceso. Sin embargo, la implementación real de un fork en el S.O. no produce duplicación del segmento de texto ya que, como se ha visto, el código es reentrante.

En realidad, la optimización del diseño de UNIX ha ido un paso más allá incorporando una técnica conocida como copy on write (copia al escribir). Esta técnica consiste en ¡no duplicar tampoco el segmento de datos!, hasta que se realice una modificación o escritura de uno de los datos, momento en el que ¡tampoco se duplicará el segmento entero! sino únicamente la página que contiene el dato a modificar. De esta forma ¡no se duplica prácticamente nada! Aunque, eso sí, el kernel del sistema deberá incorporar nuevas estructuras de datos internas para gestionar estas técnicas, lo que es objeto de estudio de los diseñadores de S.O.

EJECUCIÓN DE PROGRAMAS: exec

Con la denominación común de exec se comprende un conjunto de llamadas al sistema basadas en la primitiva execve. Esta llamada provoca el recubrimiento del programa llamante con el código de otro y el inicio de su ejecución. Por ello, cambia el programa, pero no cambia el entorno, al contrario que con un fork que funcionaba a la inversa. Por tanto, exec es "la llamada que nunca retorna". Claro, siempre que no se produzca un error, en cuyo caso retorna un valor -1. La forma general es:
int execve (nombre, argv, envp);
donde nombre es un puntero a char; argv un puntero a los argumentos y envp otro a valores de entorno (char *nombre, *argv[], *envp[]).

El nombre es el path absoluto de un ejecutable, ya sea binario o fichero de texto ejecutable. Este último, como se describió en los capítulos dedicados a shells, deberá comenzar con una línea:
#! intérprete [opciones]

En este último caso, el sistema operativo realizará un execve del intérprete para ejecutar el fichero.

Respecto al mantenimiento del entorno, es prácticamente total. El nuevo programa mantiene incluso el PID del que lo lanzó. Los descriptores de este último pasarán al nuevo abiertos (salvo que se haya utilizado el flag close-on-exec a través de la llamada ioctl). Las señales que el proceso llamante ignoraba se seguirán ignorando; sin embargo, las que interceptaba quedarán restituídas a su valor original. Del mismo modo la pila de señales recibidas y no procesadas se perderá.

PARADIGMA DE GENERACIÓN DE PROCESOS UNIX

El paradigma de generación de procesos UNIX consiste en la combinación de primitivas fork y exec. Así como

otros sistemas operativos como VMS realizan la generación de nuevos procesos con una única llamada al sistema la idea de UNIX es:

```
if ((pid = fork()) == 0) {
  /* Yo, que soy el hijo del fork, no vivire
 mucho, solo he nacido para preparar la
      venida al mundo de otro proceso.
Preparo el entorno, abro los descriptores
      necesarios, etc, etc...
  */
  execv(...);
  /* Si aqui sigo vivo es que exec fallo */
  _exit(1);
};
/* Codigo del padre */
```

En la figura 4 se ha representado uno de los ejemplos más comunes: el

Existe una zona denominada u_area con información adicional sobre cada proceso

mecanismo de generación de una shell desde el proceso 1, init. Asimismo, la figura 6 representa un simple programa de ejemplo y el modo de razonar sobre él.

En cualquier caso, el paradigma no es de uso obligado ya que se puede realizar la generación en un solo paso mediante la llamada al sistema vfork.

SINCRONIZACIÓN CON LOS HIJOS. LIMPIEZA DE ZOMBIES: wait

Tras la utilización de un fork, es posible que el proceso padre desee esperar la terminación de algún hijo antes de proseguir su ejecución. De esta forma, la propia muerte de un hijo puede servir como punto de sincronización. Para ello existe la llamada al sistema wait que se utiliza de la forma:

```
int pid;
union wait status;
...
pid = wait(&status);
```

El funcionamiento de wait es el siguiente:

a) En caso de que no existan procesos hijos, devuelve un -1. b) Si hay varios hijos ya muertos como zombies, selecciona uno cualquiera, libera su estructura de datos de la tabla de procesos y devuelve su información de status y su pid.

c) Si hay hijos pero estan vivos, espera que se muera uno y actúa como en el caso 2. En este caso, wait es una llamada bloqueante.

NOTA: Consulte en el manual en línea las llamadas wait3 y wait4.

TERMINACIÓN DE PROCESOS: exit

La terminación de un proceso UNIX se debe a una de las tres causas siguientes:

1. El programa completa todas sus instrucciones y la función main retorna.
2. El programa llama a exit.
3. Se produce un error en la ejecución o una señal.

La llamada exit acepta un parámetro: exit(int codigo); que será el código de retorno del programa. Por convenio, se suele indicar con 0 la terminacion normal y con otro valor, una terminación anormal. Tras la llamada al sistema, el proceso finaliza, y el S.O. cierra todos los descriptores que este tenía abiertos. En caso de que su padre esté esperando con un wait o atendiendo a la señal SIGCHILD, se le notifica y si el proceso que termina tiene hijos, su PPID (PID del padre) pasa a ser 1 (proceso init). La llamada exit, ademas, borra los buffers E/S del proceso, que pueden ser compartidos, por lo que si no se desea que esto ocurra, se puede utilizar _exit.

COMUNICACIÓN ENTRE PROCESOS: pipe

Como se había visto hasta ahora desde el nivel de shell, un pipe es un canal de E/S directo en memoria entre dos procesos en la misma máquina. Para generarlo desde programa, se dispone de la llamada al sistema pipe, que establece un pipe y dos descriptores asociados:

```
int filedesc[2];
pipe(filedesc);
```

Sin embargo, esto de poco sirve ya que lo que se pretende es comunicar dos procesos y, en caso de que otro proceso realizara la misma acción, crearía su propio pipe con sus propios descriptores. No obstante, con la combinación de las cuatro llamadas al sistema de control de procesos se dispone de un conjunto completo que permite realizar cualquier cosa. Así pues, la idea consiste en:

1. Generar un pipe.
2. Utilizar un fork que clonará el proceso compartiendo los descriptores y quedando ambos, padre e hijo, unidos como siameses a través del pipe.
3. Un proceso lee del pipe y el otro escribe en él como si ambos lo hicieran sobre un fichero.

Esta técnica recibe el nombre de Mecanismo de Ken Thompson, uno de los dos padres de UNIX y se puede observar en la figura 5 y en el listado:

```
int filedesc[2];
pipe(filedesc);
if (fork()==0) {
  close(filedesc[0]);
  /* El hijo escribira
     ya que cierra lectura */
};
close(filedesc[1]);
/* El padre leera
  ya que cierra escritura */
```

Con las llamadas expuestas, se dispone ya de las herramientas necesarias para control de procesos en UNIX que realmente es la parte más característica de este S.O. Como se puede apreciar, el sistema no sólamente proporciona grandes facilidades para la programación concurrente sino que, además parece invitar a ella. En los siguientes números se proporcionarán los otros dos pilares de la programación en UNIX: el manejo de la información en masa a través del sistema de ficheros y sus API y la comunicación entre procesos, ya sea a nivel de la misma máquina o de toda la internet, a través del IPC de sockets de Berkeley lo que servirá como base para el desarrollo de aplicaciones cliente-servidor.

LA TRANSFORMADA DISCRETA DE FOURIER



Juan Ramón Lehmann

Empezaremos con el análisis de la frecuencia: las imagenes pueden ser representadas en dos dominios, el espacial y el de las frecuencias, en cuyo caso (el dominio de las frecuencias) la imagen es descompuesta en una serie de senos y cosenos de diferentes frecuencias.

Básicamente la transformada de Fourier es el proceso de transformar una imagen del dominio espacial a la misma en el dominio de las frecuencias.

La transformación de Fourier, es un tema amplio y que exige tener bien claros algunos conceptos que se van a tocar brevemente.

LA LUMINANCIA

La luminancia o energía emitida por una unidad de área en una unidad de ángulo sólido, es la magnitud fotométrica que se corresponde con la sensación de claridad o brillo percibida por el ojo. El brillo es una respuesta a un estímulo externo (la longitud de onda) de nuestro ojo. Según la intensidad del brillo, se puede representar como

$$f(x,y)=\int_0^{\infty} I(x,y,\lambda)\times V(\lambda)\times d\lambda$$

Figura 1.

* donde V(y) representa la función eficiencia relativa de la luminancia.

La respuesta del ojo a la intensidad lumínica, no es lineal como cabe esperar, sino que es casi logarítmica con respecto a esta.

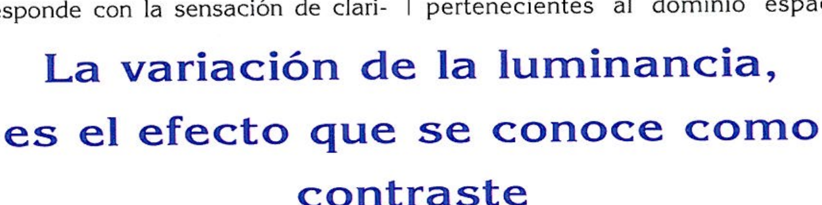
La variación de la luminancia, es el efecto que se conoce como contraste.

DOMINIO ESPACIAL Y DOMINIO DE LA FRECUENCIA

Las técnicas más acertadas a la hora de realizar mejoras de la imagen pasan por la modificación de la luminancia de un pixel teniendo en cuenta la luminancia de su entorno.

La mayor parte de las técnicas orientadas a la mejora de la imagen se desarrollan en el dominio espacial o dominio de la frecuencia.

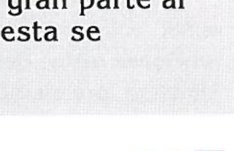
Las técnicas que modifican directamente el valor de la luminancia del pixel, se encuadran en las técnicas pertenecientes al dominio espacial. Las que modifican indirectamente la luminancia del pixel, ya sea por influencia del entorno o como resultado de una ponderación de los valores de luminancia de dicho entorno, se encuadran en el dominio de la frecuencia.



LA TRANSFORMADA DE FOURIER

Toda señal puede ser representada mediante su espectro en el dominio de las frecuencias (en el caso que nos

La convolución en el dominio del espacio, plantea muchos inconvenientes y limitaciones a la hora de filtrar una imagen debido en gran parte al medio en que esta se desarrolla.



ocupa, la imagen digital, existe una simetría perfecta entre esta y su espectro de frecuencias espaciales).

La frecuencia espacial es una distribución espacial de iluminaciones que sigue un patrón sinusoidal. La transformada de Fourier es una forma matemática de representar este fenómeno físico.

El análisis de Fourier se encargará de determinar la amplitud y la fase para los sinusoidales. El acto de juntar todas esas frecuencias para formar una sola que se adapte, se la denomina síntesis de Fourier. Estas dos operaciones básicas son posibles gracias a las fórmulas de transformación de Fourier:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{+i2\pi ux} du$$

Figura 2.

donde i=raíz cuadrada -1.

$$e^{\pm i 2\pi ux} = \cos 2\pi ux \pm i \sin 2\pi ux$$

Figura 3.

La definición de la transformada de Fourier dada en la figura 2, es válida para cualquier función integrable f(x). Esta integración descompondrá f(x) en una suma de exponenciales complejos. La función compleja F(u), especifica, para cada frecuencia <u>, la fase y la amplitud de cada exponencial. A F(u) lo llamaremos el espectro de la frecuencia.

Es importante resaltar que f(x) y F(u) son distintas representaciones de una misma función. En particular, f(x) es la señal en el dominio del espacio y F(u) la misma señal pero en el dominio de la frecuencia.

La conversión de una señal del dominio del espacio al dominio de la frecuencia, se denomina Transformada de Fourier, y el paso del dominio de la frecuencia al dominio del espacio, Transformada inversa de Fourier.

Aunque f(x) puede ser cualquier señal, solamente se contemplaran las funciones reales (las imágenes de color standard, por ejemplo).

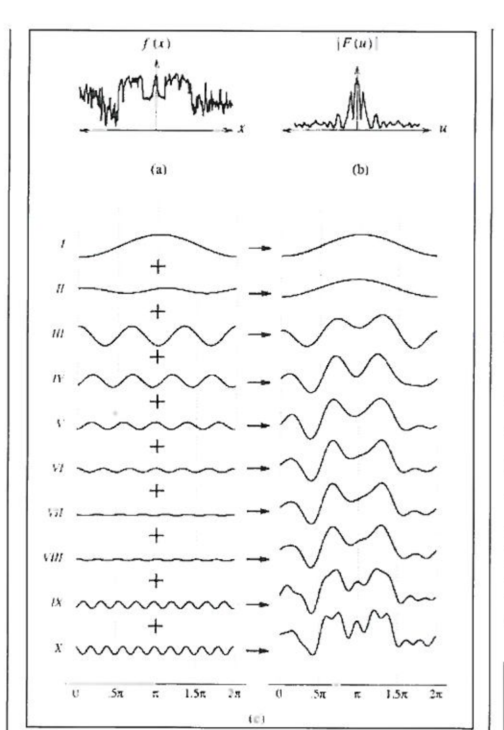


Figura 4.

Siguiendo la definición anterior, las funciones reales se codificarían como una serie de senos y cosenos, los cuales todos juntos nos darían la fase y la amplitud de cada componente en la frecuencia. Se define F(u) como una función compleja de la forma R(u)+I(u), donde R(u) y I(u) son los componentes real e imaginarios de la función respectivamente.

La amplitud o magnitud de F(u) se define como:

$$|F(u)| = \sqrt{R^2(u) + I^2(u)}$$

Figura 5.

Es a menudo llamado el Espectro de Fourier, el cual no se debe confundir con la transformada de Fourier F(u), que se conoce como el espectro.

La fase del espectro se define como:

$$\Phi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right]$$

Figura 6.

Esta especifica el ángulo de fase para el exponencial complejo de cada frecuencia <u>. La transformada de Fourier es frecuentemente tratada como una magnitud en vez de una frecuencia, ignorando el ángulo de la fase. Esta forma de visualización de los datos ha llegado a ser muy convencional debido a que el grueso de la información que acompaña a la señal, esta incluido dentro de la frecuencia, dado como la magnitud del espectro |F(u)|. Esto se puede comprender fácilmente observando la Figura 4.

Como se puede observar la frecuencia resultante es una suma (en el caso de ejemplo en la figura 4) de 5 frecuencias individuales dadas por los correspondientes senos. Al conjunto de estas frecuencias individuales, se las denomina las Series de Fourier. Las Series de Fourier se pueden expresar matemáticamente como:

$$f(x) = \sum_{n=-\infty}^{\infty} c(nu_0) e^{i2\pi nu_0 x}$$

Figura 7.

Donde c(nu0) es el n coeficiente de Fourier.

$$c(nu_0) = \int_{-x_0/2}^{x_0/2} f(x) e^{-i2\pi nu_0 x} dx$$

Figura 8.

y u0 es la frecuencia fundamental. Nótese que f(x) es periódica, por lo que la integral de la figura 8, usada para calcular el coeficiente de Fourier, debe ser integrada en un período de x0.

LA SEPARABILIDAD DE LA TRANSFORMADA DE FOURIER

Una cualidad importante de la transformada de Fourier es su característica de separabilidad (suponiendo que disponemos de una imagen cuadrada N*N). Por separabilidad se entiende la posibilidad de escribir la Transformada de Fourier como:

$$F(u,v) = \frac{1}{N} \cdot \sum_{x=0}^{N-1} e^{-j\frac{2\pi ux}{N}} \cdot \sum_{y=0}^{N-1} f(x,y) \cdot e^{-j\frac{2\pi vy}{N}}$$

Figura 9.

Esta nos permite calcular la transformada en dos pasos, primero las filas y luego las columnas. Lo cual nos

dejaría una transformada unidimensional que reescribiríamos como sigue:

$$F(x,v)=\sum_{y=0}^{N-1} f(x,y)\cdot e^{j\frac{2\pi vy}{N}}$$

Figura 10.

Se incluye en el disco que acompaña a la revista un código fuente que calcula la transformada discreta de Fourier de una matriz de imagen cuadrada.

CÁLCULO DE LA TRANSFORMADA INVERSA

En principio, cualquier algoritmo utilizado para calcular la transformada de Fourier en una serie de valores discretos, puede utilizarse con muy pocos cambio para calcular la transformada inversa de Fourier. Partiendo de que la transformada directa la definimos con la ecuación:

$$F(u)=\frac{1}{N}\cdot\sum_{x=0}^{N-1} f(x)\cdot e^{-j\frac{2\pi ux}{N}}$$

Figura 11.

y la inversa con:

$$f(x)=\sum_{u=0}^{N-1} F(u)\cdot e^{j\frac{2\pi ux}{N}}$$

Figura 12.

Calculando la conjugada compleja de la inversa y dividiendo ambos miembros de la misma por N, nos resulta:

$$\frac{1}{N}f^*(x)=\frac{1}{N}\sum_{u=0}^{N-1} F^*(u)\cdot e^{-j\frac{2\pi ux}{N}}$$

Figura 13.

Esto nos lleva a que si le da a como argumento el array de valores en el dominio de Fourier, previamente conjugados, nos devolverá la secuencia de valores en el dominio espacial conjugada y dividida por N(f*(x)/N). Si reescribimos la ecuación para los casos bidimensionales tendríamos:

$$f^*(x,y)=\frac{1}{N}\cdot\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F^*(u,v)\cdot e^{-j\frac{2\pi(ux+vy)}{N}}$$

Figura 14.

DISEÑO DE FILTROS EN EL DOMINIO DE LA FRECUENCIA

Una vez tenemos diseñado la máscara de un filtro, basta con multiplicarlo por la transformada de Fourier de la señal a filtrar. Se ha de tener las siguientes consideraciones antes de realizar los filtrados:

- Una imagen en el dominio del espacio es finita, no así esa misma imagen en el dominio de las frecuencias. Al limitar la ventana ya de por si se eliminan las frecuencias altas de la señal, por lo que se debe de procurar que la imagen a filtrar ocupe la mayor parte posible de la ventana para que su espectro se reduzca al máximo.

- Normalmente el filtro consta de una parte real y otra imaginaria, normalmente se limitará a la parte real.

- Con este sistema se puede implementar en principio cualquier filtro lineal.

Una vez tenemos diseñado la máscara de un filtro, basta con multiplicarlo por la transformada de fourier de la señal a filtrar

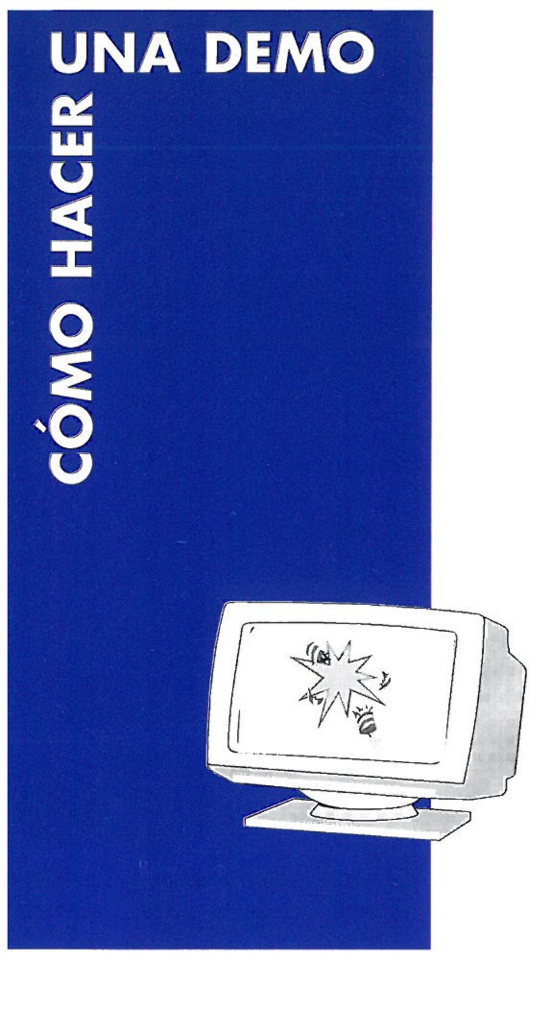
Una vez que se dispone de la transformada del filtro a aplicar (o lo que es lo mismo, el filtro en el dominio de las frecuencias), se puede decidir si se desea filtrar en el dominio del espacio con el teorema de la convolución (calculando la transformada inversa de Fourier de la máscara del filtro, dando así la máscara de convolución espacial) o si por el contrario si se desea transformar la señal de la imagen al dominio de las frecuencias.

La decisión de utilizar un sistema u otro depende altamente del tamaño de la imagen y de la máscara de filtro a utilizar. Cualquier filtro linear es directamente utilizable en el dominio de las frecuencias, y su precisión puede ser tan alta como el programador desee (si se dispone de una ventana lo suficientemente grande). La mayor pega reside en el tiempo y espacio que requiere el dominio de las frecuencias frente al dominio del espacio.

RESUMEN

La Transformada de Fourier es un tema bastante amplio y muy apasionante, temas sin tocar en el presente articulo como la implementación de la transformada inversa de Fourier (IFT), las aplicaciones reales de la misma, y por supuesto los ejemplos prácticos de todo esto, son temas a tratar en el próximo capítulo.

Algunos términos que aparecen en esta entrega se dan por sabidos, de cualquier manera se encuentra la explicación en capítulos anteriores. Se recomienda repasar la entrega del número 10.



LOS TEXTOS

Pedro Antón

Probablemente este sea el artículo menos "demoero" de los vistos hasta ahora pero no por ello el menos importante. ¿Qué tipo de demo es aquella en la que no se lee (o por lo menos se intenta leer) algún texto?. Es de una importancia vital hacer llegar al espectador algún tipo de mensaje, por ejemplo quien es el grupo tan fabuloso que ha invertido un montón de horas de su tiempo libre en hacer lo que se está viendo, o cual es el nombre de los increíbles "coders" del grupo, sin dejar a un lado los nombres de los Jean Michael Jarre particulares de cada grupo y sin olvidarse tampoco de los grafistas. En otras ocasiones se indicará el nombre de algún efecto o se comentará algo en particular de alguno, pero siempre será necesario escribir algo en la pantalla del ordenador.

El código que acompaña a este artículo es más útil para hacer una "intro", que para hacer una demo, ya que aprovecha la definición de las fuentes del sistema para ahorrar memoria, una intro no puede sobrepasar nunca los 4 Kb, no obstante el proceso para definir una fuente "demoera" es bastante similar, además con unas ligeras ampliaciones, es posible definir una fuente de usuario que use máscara de color consiguiendo así una fuente bastante interesante de usar en una demo sin excesivas complicaciones.

Toda demo lleva un mensaje implícito, es totalmente imprescindible poner en la pantalla algún texto para llamar la atención del espectador de la demo.

LOS COMANDOS

Más comandos, más versatilidad.

Una buena fuente, ha de tener comandos, entendiéndose por estos secuencias de bytes no legibles que varíen el formato de escritura. El código del artículo tiene únicamente cuatro comandos; final del *string*, tipo de letra, color, y *bold*, el lector podrá añadir tantos como se le ocurran.

Fin de cadena de texto. Una byte de valor cero indicará que la cadena de texto ha finalizado, por tanto no hay que seguir escribiendo.

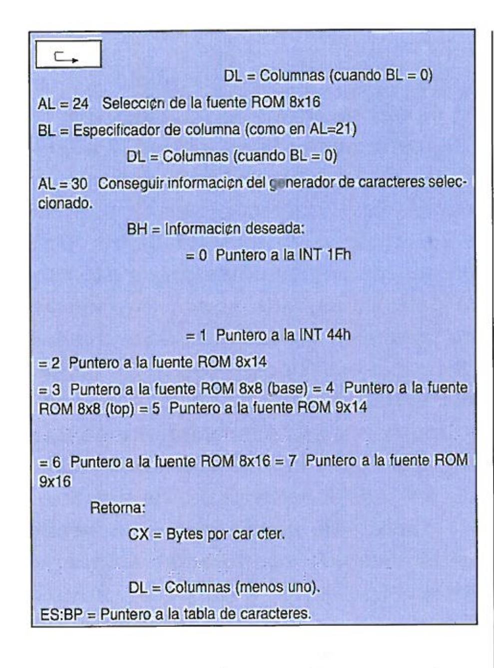
-Tipo de letra.

Este comando indica a la rutina principal donde debe buscar la definición de las fuentes en la memoria del ordenador, El ejemplo usa tres tipos de fuentes: 8x8, 8x16 y definida por el usuario. El servicio 11h de la interrupción 10h devuelve en ES:BP la dirección donde se encuentra almacenada la fuente seleccionada en BH.

SERVICIO 11h DE LA INTERRUPCION 10h.

```
Llamar a la interrupci¢n 11 con las siguientes entradas:
AH = 11h ( Servicio 11h)
AL = 00 Carga de una tabla de caracteres de usuario
            BH = N£mero de bytes por car cter.
            BL = N£mero de tabla.
            CX = Cantidad de caracteres en la tabla.
DX = C¢digo ASCII del primer car cter definido.
     ES:BP = Puntero a la tabla caracteres de usuario.
AL = 01   Selecci¢n de la fuente ROM BIOS 8x14.
            BL = N£mero de tabla en el generador de caracteres
RAM
AL = 02   Selecci¢n de la fuente ROM BIOS 8x8.
            BL = N£mero de tabla en el generador de caracteres
RAM
AL = 03   Selecciona la tabla de definici¢n de fuente.
BL = Valor del registro del selecci¢n de mapa de
                        caracteres (EGA/VGA).
= Numero de tabla del generador de caracteres RAM (MCGA).
AL = 04   Selecci¢n de la fuente ROM BIOS 8x16.
            BL = N£mero de tabla en el generador de caracteres
RAM.
AL = 10   Carga de una tabla de caracteres de usuario.
            BH = N£mero de bytes por car cter.
            BL = N£mero de tabla.
            CX = Cantidad de caracteres en la tabla.
DX = C¢digo ASCII del primer car cter definido.
     ES:BP = Puntero a la tabla caracteres de usuario.
AL = 11   Selecci¢n de la fuente ROM BIOS 8x14.
            BL = N£mero de tabla en el generador de caracteres
RAM
AL = 12   Selecci¢n de la fuente ROM BIOS 8x8.
            BL = N£mero de tabla en el generador de caracteres
RAM
AL = 14   Selecci¢n de la fuente ROM BIOS 8x16.
            BL = N£mero de tabla en el generador de caracteres
RAM.
AL = 20   Puntero a la tabla de caracteres de usuario para la inte-
rrupci¢n 1Fh (8x8).
     ES:BP = Puntero a la tabla de usuario.
AL = 21   Puntero a la tabla de caracteres de usuario para la inte-
rrupci¢n 43h.
            BL = Especificador de columnas.
= 0 - Especificadas por el usuario  (DL = columnas). = 1 - 14 colum-
nas.
                  = 2 - 25 columnas.
                  = 3 - 43 columnas.
            CX = Bytes por car cter.
            DL = Columnas (cuando BL = 0)

     ES:BP = Puntero a la tabla de usuario.
AL = 22   Selecci¢n de la fuente ROM 8x14
BL = Especificador de columnas (como en AL=21)
            DL = Columnas (cuando BL = 0)
AL = 23   Selecci¢n de la fuente ROM 8x8
BL = Especificador de columnas (como en AL=21)
```



DL = Columnas (cuando BL = 0)

AL = 24 Selecci¢n de la fuente ROM 8x16

BL = Especificador de columna (como en AL=21)

DL = Columnas (cuando BL = 0)

AL = 30 Conseguir informaci¢n del generador de caracteres seleccionado.

BH = Informaci¢n deseada:

= 0 Puntero a la INT 1Fh

= 1 Puntero a la INT 44h

= 2 Puntero a la fuente ROM 8x14

= 3 Puntero a la fuente ROM 8x8 (base) = 4 Puntero a la fuente ROM 8x8 (top) = 5 Puntero a la fuente ROM 9x14

= 6 Puntero a la fuente ROM 8x16 = 7 Puntero a la fuente ROM 9x16

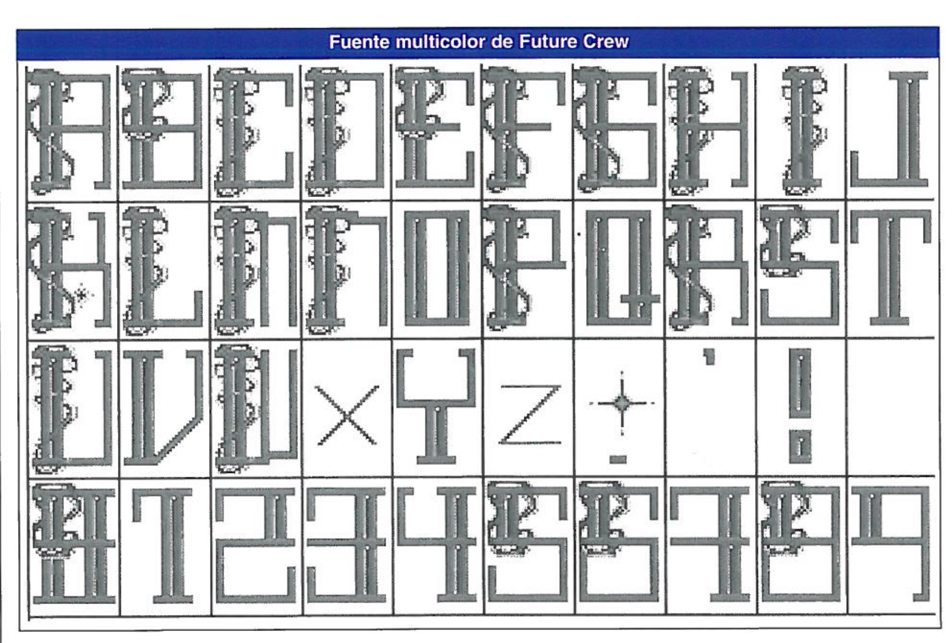
Retorna:

CX = Bytes por car cter.

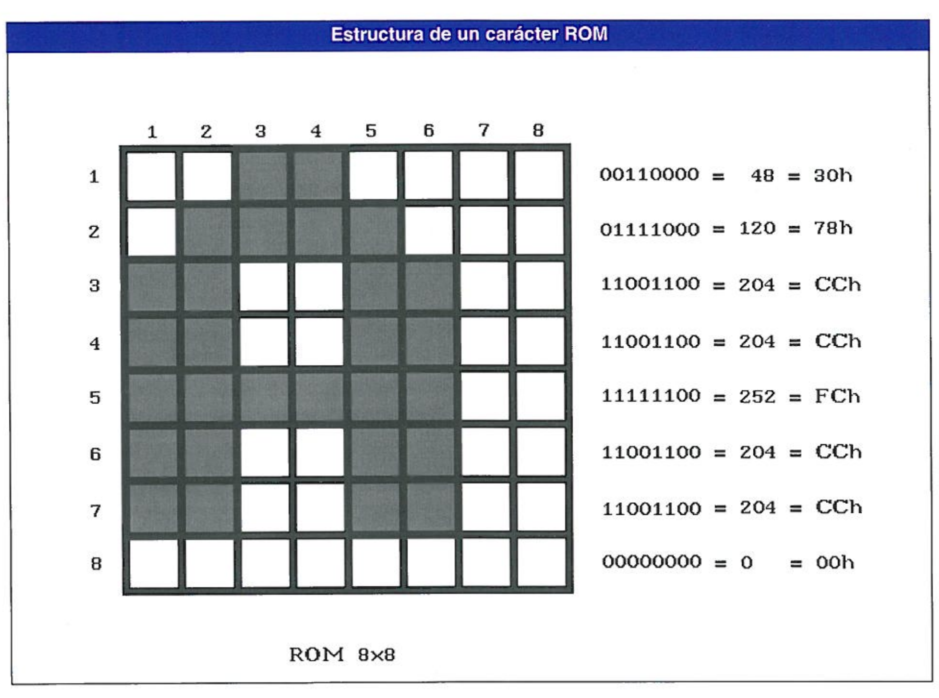
DL = Columnas (menos uno).

ES:BP = Puntero a la tabla de caracteres.

Fuente multicolor de Future Crew



Estructura de un carácter ROM



Las fuentes de usuario son perfectamente controlables por el programador y en cualquier momento se puede conocer su dirección.

El concepto de fuente de usuario es algo que hay que prestar atención a la hora de diseñar un código multiuso, se debe tener en cuenta que el tamaño de dicha fuente es definible, por tanto se debe almacenar en algún lugar de la memoria el ancho y alto de la fuente que se este usando, aunque parte del código adjunto presupone que el ancho de la fuente es ocho.

-Color.

Este comando junto con la definición de los caracteres es el que puede parecer más pobre a la hora de usar este tipo de fuente en una demo, ya que en principio puede hacer creer que la fuente sólo puede tener un color, como ocurre en el ejemplo, sin embargo eso siempre queda a merced del programador que siempre podrá ingeniárselas para aplicar múltiples colores a esa fuente de distintas formas.

-Bold.

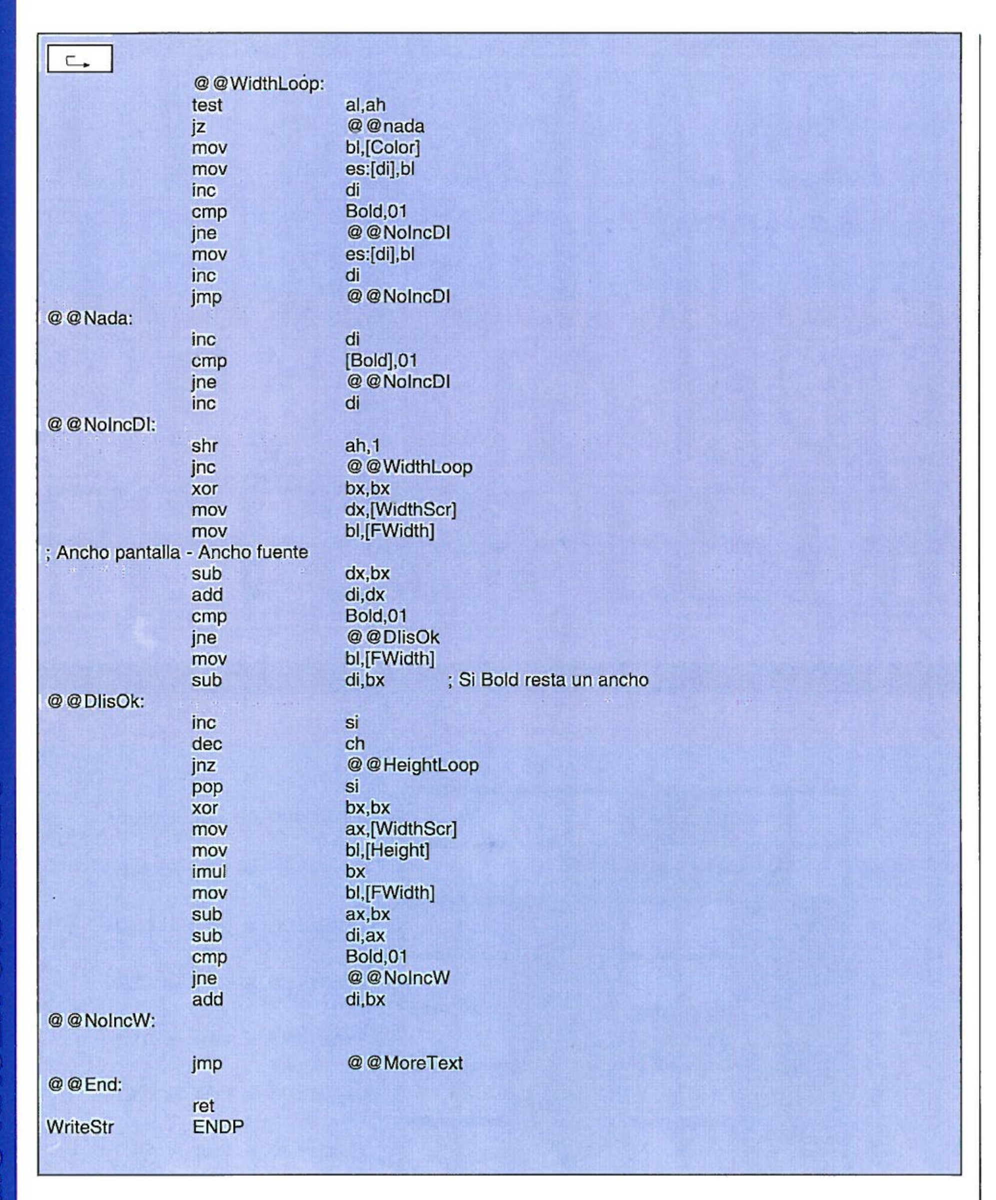
El comando bold escribe una fuente el doble de ancha, esto es, poner dos bytes por cada uno. En lo que respecta a comandos similares, el lector puede intuir la sencillez de hacer una fuente el doble de alta, implementando de esta manera el comando *thin*, o inclinar ligeramente la fuente para obtener un comando *italic* que escriba la fuente en cursiva. De esta forma se pueden implementar tantos comandos como se desee. En el ejemplo adjunto podrá haber un máximo de 32 comandos, que es el número de códigos ASCII que en principio, no tienen representación en pantalla.

COMO SE ALMACENA UNA FUENTE

Cada bit indica si se ha de poner, o no, un punto. Las fuentes del sistema están almacenadas en determinadas direcciones de la memoria ROM, esta dirección no es siempre la misma, por ello, en primer lugar debemos informarnos de forma adecuada (servicio 11h de la INT 10h) de estas direcciones, ahora bien, será necesario saber como está almacenada esa información. En el caso de las fuentes del sistema, como son monocromas, la información que almacena cada punto es un bit, por tanto un byte serán 8 puntos de la fuente.

Esto conlleva un pequeño problema a la hora de programar la rutina que imprima las fuentes en pantalla, ya que se deberá comprobar cada bit en cada byte leído, sin embargo en una fuente con 256 colores cada punto será un byte.

En este caso, la fuente será almacenada por el programador de la forma

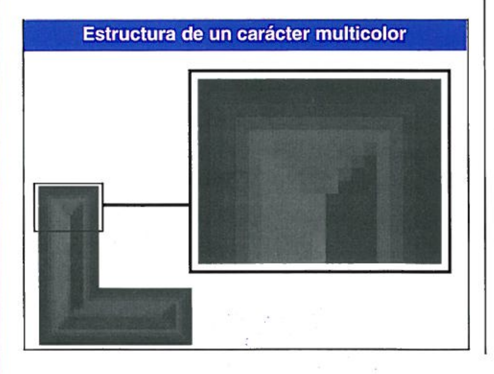


```
            @@WidthLoop:
            test        al,ah
            jz          @@nada
            mov         bl,[Color]
            mov         es:[di],bl
            inc         di
            cmp         Bold,01
            jne         @@NoIncDI
            mov         es:[di],bl
            inc         di
            jmp         @@NoIncDI
@@Nada:
            inc         di
            cmp         [Bold],01
            jne         @@NoIncDI
            inc         di
@@NoIncDI:
            shr         ah,1
            jnc         @@WidthLoop
            xor         bx,bx
            mov         dx,[WidthScr]
            mov         bl,[FWidth]
; Ancho pantalla - Ancho fuente
            sub         dx,bx
            add         di,dx
            cmp         Bold,01
            jne         @@DIisOk
            mov         bl,[FWidth]
            sub         di,bx        ; Si Bold resta un ancho
@@DIisOk:
            inc         si
            dec         ch
            jnz         @@HeightLoop
            pop         si
            xor         bx,bx
            mov         ax,[WidthScr]
            mov         bl,[Height]
            imul        bx
            mov         bl,[FWidth]
            sub         ax,bx
            sub         di,ax
            cmp         Bold,01
            jne         @@NoIncW
            add         di,bx
@@NoIncW:

            jmp         @@MoreText
@@End:
            ret
WriteStr    ENDP
```

que mejor se adapte a sus necesidades, Future Crew en su segunda megademo, UNREAL, almacena sus fuentes en forma de gráfico, capturando cada carácter como si de un *sprite* se tratara y procesando esos *sprites* correctamente.

Para realizar una fuente de este tipo se deben tener en cuenta los siguientes factores:


Estructura de un carácter multicolor

- Ancho de la letra.
- Alto de la letra.
- Ancho de la pantalla o del buffer donde vamos a trabajar.

Cada bit indica si se ha de poner, o no, un punto

- ***Color Base.***

El ancho lógico de la pantalla o del *buffer* es especialmente importante cuando se realiza un *scroll* o al trabajar en *xmode* donde se suele trabajar con formatos de ancho físico 320 *pixels*, y ancho lógico 1280 *pixels*, por ejemplo. Supongamos un modo de trabajo *xmode* donde la pantalla física es de 320x200, pero la pantalla lógica es de 1280x200. Cuando la rutina escriba las fuentes lo hará en la pantalla lógica, por lo tanto esta rutina debe saber que cantidad debe sumar al último *pixel* escrito para pasar a la línea inmediatamente inferior y continuar así escribiendo el carácter en curso. Es interesante incorporar el color base a la hora de definir una fuente, ya que esto nos permitirá, con la paleta adecuada, variar los colores de la fuente. No obstante debe quedar claro que este tipo de cosas, no tienen por ser controladas por todos los programadores, son sencillamente pautas a seguir para la correcta elaboración de una rutina de escritura de fuentes.

Ahora sólo queda buscar la forma de almacenar cada carácter. Como he dicho en líneas anteriores, Future Crew, en sus primeras demos al menos, almacena las fuentes en forma gráfica, lo cual indica la versatilidad de cada grupo para guardar sus fuentes como al programador le parezca conveniente. Personalmente no me gusta ese sistema de almacenamiento, Spanish Lords tiene su propia forma de almacenar sus fuentes en forma de fichero SLF (Spanish Lords Font file), esto requiere la preparación de una herramienta de trabajo paralela a las demos, pero que será muy empleada, esta herramienta será la encargada de capturar los bloques de bits de una pantalla con las fuentes, y escribir los datos a un fichero de forma organizada. El fichero tendrá una cabecera de información de la fuente para que la rutina que procese esos datos conozca el ancho y alto de la fuente, así como su color base, una vez leídos esos datos se almacena la fuente en memoria y se crea una tabla con el *offset* de cada carácter, ahorrando así un precioso tiempo cada vez que se escriba un carácter.

CÓMO SE ESCRIBE UN CARÁCTER

Escribir un carácter es más fácil en una fuente multicolor, pues se lee el valor y se le suma un valor de color. En caso de contemplar la posibilidad de cambio de color, en una fuente monocroma

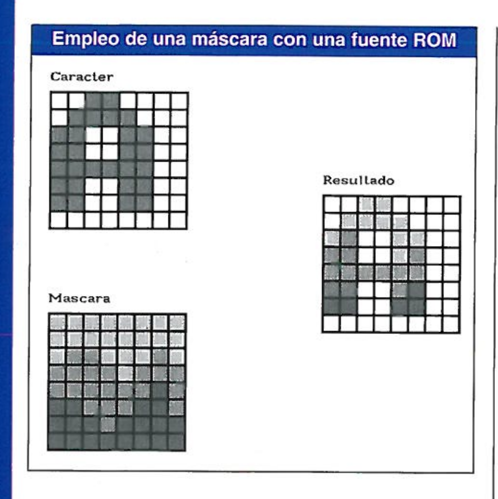
Rutina de escribir fuente monocroma

```
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ ;   WriteStr
;
;                       ES:DI- Dir donde vamos a escribir
;                       DS:SI- Dir donde est  el texto a escribir
; ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ WriteStr PROC   NEAR
@@MoreText:
                mov         cl,ds:[si]                  ; cl = primer byte
                cmp         cl,0
                je          @@End                       ; 0 = final de cadena.
                cmp         cl,3                        ; ¨Es un comando? ;-?
                ja          @@NoCommand
                mov         ch,ds:[si+1]                ; cl = comando.
                                                        ; ch = par metro.
; Si no es uno no es tipo de fuente.
                cmp         cl,1
                jne         @@NoFontType
                cmp         ch,0
                jne         @@No8x8
                mov         ax,[Seg8x8]
                mov         [SegFont],ax
                mov         ax,[Off8x8]
                mov         [OffFont],ax
                mov         [FWidth],08
                mov         [Height],08
                jmp         @@EndCommand
@@No8x8:
                cmp         ch,1
                jne         @@No8x16
                mov         ax,[Seg8x16]
                mov         [SegFont],ax
                mov         ax,[Off8x16]
                mov         [OffFont],ax
                mov         [FWidth],08
                mov         [Height],16
                jmp         @@EndCommand
@@No8x16:
; Ahora no compruebo, pero puedo ampliar comandos ¨no?
;               cmp         ch,2
;               jne         @@NoUsrFont                 ;-)
                mov         ax,[SegUsrFont]
                mov         [SegFont],ax
                mov         ax,[OffUsrFont]
                mov         [OffFont],ax
                mov         al,WUsrFont
                mov         [FWidth],al
                mov         al,HUsrFont
                mov         [Height],al
                jmp         @@EndCommand
@@NoFontType:
                cmp         cl,2                        ; Si no es dos no es color.
                jne         @@NoColor
                mov         [Color],ch
                jmp         @@EndCommand
@@NoColor:
                cmp         cl,3                        ; Si no es tres no es Bold.
                jne         @@NoBold
                mov         [Bold],ch
                jmp         @@EndCommand
@@NoBold: ;°°°°°°°°°°°°°°° ;   Aqu¡ estar  la implementaci n del resto de comandos :) ;°°°°°°°°°°°°°°°°°°°°°°°°°°°°

@@EndCommand:
                add         si,2        ; Un comando son dos bytes ;)
                jmp         @@MoreText
@@NoCommand:
; Listo para el siguiente caracter :-O

                inc         si
                push        si          ; Guarda el puntero.
                mov         si,[OffFont]
                mov         ax,[SegFont]
                mov         fs,ax
                xor         ax,ax
; CL = Caracter a poner.
                mov         al,cl

                mov         ch,[Height]
; Codigo ASCII * Altura de caracter.
; 100 % OPTIMIZABLE
                imul        ch
                add         si,ax       ; si = caracter.
@@HeightLoop:
                mov         al,fs:[si]
                mov         ah,10000000b
```



debemos leer cada bit y poner un byte de color por cada bit a uno. Al finalizar el ancho de un carácter, se suma el valor correspondiente a la diferencia entre el ancho de cada carácter y el de la pantalla lógica y hecho.

Queda por explicar cómo se puede aplicar una máscara de color a una fuente monocroma. Con esto se puede conseguir una fuente multicolor con algún efecto original, sin sacrificar memoria, muy útil para una "intro", como ya se comentó al principio del artículo. Esto se consigue definiendo un dibujo del tamaño de la fuente que se vaya a emplear, y empleando ese dibujo a modo de máscara: si hay que poner un *pixel* en pantalla (o en el *buffer*) se pondrá del color que indique la máscara, si a esto se le añade un efecto de paleta se consiguen efectos realmente asombrosos con muy pocas líneas de código.

La paleta puede crear bonitos textos

El buffer se usa para hacer scrolls

LA VELOCIDAD

El código que acompaña al artículo es bastante fácil de entender y controla varios factores, como distintos anchos de fuente y de pantalla lógica, pero evidentemente esta versatilidad se paga, y el código no es todo lo rápido que debiera. Siempre que sea posible se debe evitar el uso de la instrucción IMUL, y este código la emplea innecesariamente. Una buena optimización se conseguirá fijando siempre el ancho cada carácter a ocho.

LA PALETA

Se pueden crear bonitos efectos, implementando el uso de una máscara al código adjunto (se hará necesario el uso de un comando de color similar al de una fuente multicolor) consiguiendo así fuentes de distintos colores. Si además el lector a comprendido la forma de tratar la paleta, se pueden conseguir efectos con los textos, bastante llamativos.

EL BUFFER

Este artículo no trata ningún efecto, luego el buffer dependerá del efecto al que unamos la rutina, no obstante añadiendo a la rutina un buffer ancho se conseguirá con facilidad un *scroll* por soft. Mas adelante se verá que se pueden realizar *scrolls* hardware, siendo estos mucho más rápidos, además de consumir mucho menos tiempo de procesador.

EL BUCLE PRINCIPAL

La complejidad del bucle principal depende del número de comandos. Lo primero que se debe hacer es comprobar si el byte leído es de control o por el contrario se trata de un carácter que hay que escribir.

En caso de tratarse de un comando se han de leer dos bytes, el primero indicará el número de comando y el siguiente el parámetro de dicho comando, a continuación se procede a interpretar el comando y a procesar su función, actualizando las variables que sea necesario.

Si el byte no es un comando, es un carácter que se debe escribir en la zona de memoria que indique ES:DI. Ahora, se debe tener en cuenta el ancho de la zona de memoria a escribir, el ancho de la fuente, y el tipo de parámetro que se encuentre activo, ya que si el comando *bold* está activo, el ancho de la fuente es justo el doble. Una vez visto esto se buscarán los bytes donde se encuentra definido el carácter, multiplicando el código ASCII, o valor del byte a poner, por el alto de la fuente, y sumando ese valor a la dirección de memoria donde están definidas las fuentes. Hay que leer de aquí tantos bytes como tenga la fuente de alto e interpretarlos de la forma correcta:

Como se ha dicho anteriormente, en este tipo de fuente cada bit representa un punto en la pantalla, luego se deberá buscar una forma eficaz de leer cada bit del byte a tratar. Haremos una comprobación del byte a tratar con el número binario 10000000, de tal manera que obtengamos activo el *flag* de cero cuando no esté a uno el bit 7 de dicho byte. En caso contrario escribiremos un byte con el color seleccionado en la posición de memoria correspondiente. Para comprobar el siguiente bit bastará con rotar a la derecha un bit, cuando se active el *flag* de *carry* es que hemos leído los ocho bits del byte, entonces se actualizan los punteros y se pasa a la siguiente línea.

Y éste, es el cuerpo del procedimiento que escribe un carácter, espero que saquéis algo en claro de todo este lío de bits y bytes, y podáis crear una buena rutina para escribir texto en vuestras demos.

PROGRAMACIÓN DEL MIXER DE LA SOUND BLASTER

Héctor Martínez

Una de las novedades que introdujeron las tarjetas de sonido en su día, fue la incorporación de un mezclador interno (mixer), que nos permite seleccionar la fuente de entrada y los canales de salida que queremos oír, a parte de permitir ponderar a nuestro gusto el volumen de cada canal.

Hasta el momento las tarjetas Sound Blaster utilizan dos tipos de chips: CT1345 y CT1745. El CT1345 es usado en la Sound Blaster Pro y el CT1745 en la Sound Blaster 16.

Para comenzar se explicará la secuencia de programación del mixer. Esta secuencia es aplicable a la programación de los dos chips. Posteriormente se hará referencia a las características y a la función de cada registro de cada uno de los dos chips presentados anteriormente.

SECUENCIA DE PROGRAMACIÓN

Ambos chips utilizan dos direcciones de puertos consecutivos: 2x4h y 2x5h donde x depende de la dirección donde se haya configurado la dirección base de la tarjeta. El puerto 2x4h es un puerto de escritura (*write only*) y es el llamado *Address Port* (Puerto de dirección). El 2x5h es de lectura y escritura y es el *Data Port* (Puerto de datos).

La secuencia de programación del mixer es la siguiente:

1. Escribir el índice del registro del mixer al que queremos acceder en el *Address Port*.
2. Escribir o leer el contenido del registro accediendo al *Data Port*.

El siguiente código en ensamblador muestra el proceso de acceder a los registros del mixer:

```
; Código común para especificar a que registro queremos acceder
mov dx, SBBaseAddress Dirección base de la Sound Blaster
add dx,4 ; Dirección base del mixer
mov al,MixerRegIndex ; Índice del registro del mixer al que queremos acceder
out dx,al ; Selección del registro
inc dx ; Ahora dx apunta al Data Port

; Código para leer un valor del mixer
in al,dx ; En al tenemos el dato que queríamos leer

; Código para escribir un valor en el mixer
mov al, Valor
out dx,al ; Se escribe el valor deseado en el registro seleccionado
```

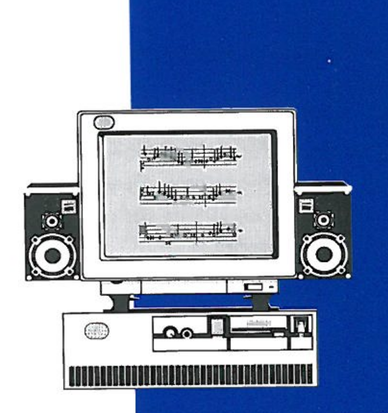
EL MIXER DE LA SOUND BLASTER PRO: EL CHIP CT1345

CARACTERÍSTICAS:

El CT1345 es un mixer estéreo que controla el volumen, la mezcla de salida y la selección de la fuente de entrada (sólo una). También controla un filtro pasa-bajos de entrada y otro de salida. Este filtro sólo se puede activar o desactivar, y no se puede controlar su ganancia.

CONTROL DE VOLUMEN:

Este chip es capaz de controlar independientemente los canales izquierdo y derecho de los canales Voice, MIDI, CD y Line-In. Y por supuesto del Master. Utilizo la nomenclatura inglesa, ya que la traducción no la encuentro apropiada. Todos estos canales tienen sólo 8 niveles de volumen, ya que se codifican utilizando 3 bits. La salida del micrófono es mono con 4 niveles. Este control, no



Prácticamente todas las tarjetas de sonido actuales disponen de un mezclador interno programable por software. En el caso de las tarjetas de la familia Sound Blaster, la primera en disponer de mezclador fue la SB Pro, y a partir de entonces lo han seguido incorporando el resto de las tarjetas de esta familia: SB 16 y SB AWE 32.

afecta al volumen de grabación, sólo al volumen de salida del micrófono.

SELECCIÓN DE LOS CANALES DE SALIDA:

La salida final a través del Line-Out es la suma de los canales Voice, MIDI, CD, Line-in, Micrófono y PC Speaker. Para dejar de oír uno de los canales, la única solución es reducir el volumen de ese canal a cero.

SELECCIÓN DEL CANAL DE ENTRADA:

Este chip sólo permite la selección de una fuente de entrada. Ésta puede ser el CD, Line-In o el Micrófono. El valor por omisión después de hacer un reset del mixer es el micrófono.

CONTROL DE LOS FILTROS DE ENTRADA Y SALIDA:

El CT1345 Dispone de un filtro pasabajos de salida de 3,2 kHz. Tiene dos filtros de entrada, uno de 3,2 kHz y otro de 8,8 kHz.

HABILITACIÓN DE LA REPRODUCCIÓN EN ESTÉREO:

Como ya se vio hace dos entregas, para que la Sound Blaster Pro reprodujera en estéreo, había que habilitar un bit del mixer. En aquel número se hacía sin saber que registro se estaba cambiando, ahora sabremos donde está situado dicho registro.

REGISTROS DEL MIXER DE LA SOUND BLASTER PRO

Ahora pasaremos a la explicación de cada uno de los registros del mixer. Para una rápida visión mirar la *tabla 1.* El lector que no entienda la ganancia en decibelios, no se tiene que preocupar, basta con que sepa que el valor 0 indica siempre silencio, y el valor máximo indica el volumen máximo.

Registro 0x00 (Reset)

Escribiendo cualquier valor en este registro, el resto de registros serán restaurados a sus valores por omisión.

Registro 0x0A (Volumen del micrófono)

2 bits, consiguiendo 4 niveles.

0-3: -46 dB a 0 dB, en saltos de aproximadamente 7 dB, excepto el 0, que pasa de golpe a -46 dB.

El valor por omisión es 0 (-46 dB).

Registro 0x0C, bits 1 y 2 (Canal de entrada)

Estos dos bits indican la fuente de entrada que se utilizará cuando se ponga la tarjeta en el modo de digitalización. Los valores son los siguientes:

0 ó 2: Selección del micrófono.

1: Selección de la entrada interna de CD.

3: Selección de la entrada de línea (Line-in).

El valor por omisión es 0: Micrófono.

Registro 0x0C, bit 3 (Filtro pasabajos de entrada)

Este registro sólo se tiene en consideración si está activo el filtro de entrada.

0: Se activa el filtro pasabajos de 3,2 kHz.

1: Se activa el filtro pasabajos de 8,8 kHz.

El valor por omisión es 0: filtro pasabajos de 3,2 kHz.

Índice	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	Reset del mixer							
0x04	Voice izquierdo				Voice derecho			
0x0A						Micrófono		
0x0C			Filtro de entrada		Tipo de filtro	Fuente de entrada		
0x0E			Filtro de salida				Estéreo On/Off	
0x22	Master izquierdo				Master derecho			
0x26	MIDI izquierdo				MIDI derecho			
0x28	CD izquierdo				CD derecho			
0x2E	Line-In izquierdo				Line-In derecho			

Tabla 1. Mapa de registros del CT1345.

Registro 0x0C, bit 5 (Filtro de entrada)

Se utiliza para activar o desactivar el filtro de entrada.

0: Filtro pasabajos de entrada activado.

1: Filtro pasabajos de entrada desactivado.

Valor por omisión 0: Filtro activado.

El filtro de entrada se utiliza para filtrar las altas frecuencias durante la grabación para obtener mejor calidad cuando la frecuencia de muestreo es baja. Se recomienda utilizar el filtro pasabajos de 3,2 kHz para grabaciones mono con frecuencias de muestreo inferiores a 18 kHz. Para grabaciones mono con una frecuencia entre 18 kHz hasta 36 kHz se recomienda utilizar el filtro pasabajos de 8,8 kHz. Desactivar el filtro para digitalizaciones con frecuencias de muestreo mayores de 36 kHz y para cualquier grabación en estéreo.

Registro 0x0E, bit 1 (Indicador del modo estéreo)

Este registro es el que activa o desactiva las reproducciones en estéreo.

0: Salida mono.

1: Salida estéreo.

El valor por omisión es 0: Salida mono.

Registro 0x0E, bit 5 (Filtro de salida)

Activa y desactiva el filtro de salida.

0: Filtro activado.

1: Filtro desactivado.

El valor por omisión es 0: Filtro activado.

Desactivar el filtro en reproducciones con frecuencias de muestreo altas o en estéreo.

Registro 0x04 (Volumen del canal Voice)

Es el sonido generado por el DSP que sale a través del DAC (Digital to Analogic Conversor). El utilizado por los sonidos digitalizados, los players...

Registro 0x22 (Volumen de Master)

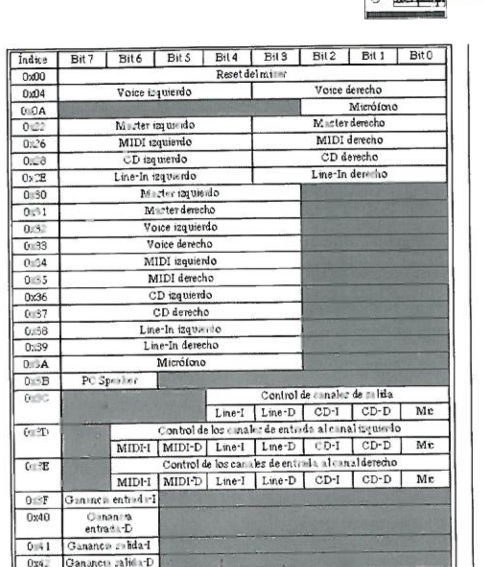
Es el volumen global de la tarjeta.

Registro 0x26 (Volumen del canal MIDI)

Es el volumen de la música generada por el chip OPL3 interno, que es el que produce la música que generalmente utilizan los juegos.

Registro 0x28 (Volumen del CD)

Sólo en el caso de disponer de un CD conectado a la tarjeta.



Índice	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	Reset del mixer							
0x04	Voice izquierdo				Voice derecho			
0x0A						Micrófono		
0x22	Master izquierdo				Master derecho			
0x26	MIDI izquierdo				MIDI derecho			
0x28	CD izquierdo				CD derecho			
0x2E	Line-In izquierdo				Line-In derecho			
0x30	Master izquierdo							
0x31	Master derecho							
0x32	Voice izquierdo							
0x33	Voice derecho							
0x34	MIDI izquierdo							
0x35	MIDI derecho							
0x36	CD izquierdo							
0x37	CD derecho							
0x38	Line-In izquierdo							
0x39	Line-In derecho							
0x3A	Micrófono							
0x3B	PC Speaker							
0x3C				Control de canales de salida				
				Line-I	Line-D	CD-I	CD-D	Mic
0x3D		Control de los canales de entrada al canal izquierdo						
		MIDI-I	MIDI-D	Line-I	Line-D	CD-I	CD-D	Mic
0x3E		Control de los canales de entrada al canal derecho						
		MIDI-I	MIDI-D	Line-I	Line-D	CD-I	CD-D	Mic
0x3F	Ganancia entrada-I							
0x40	Ganancia entrada-D							
0x41	Ganancia salida-I							
0x42	Ganancia salida-D							
0x43								AGC
0x44	Agudos izquierdo							
0x45	Agudos derecho							
0x46	Graves izquierdo							
0x47	Graves derecho							

Tabla 2. Mapa de registros del CT1745.

Registro 0x2E (Volumen del canal Line)

Controla el volumen de la señal de entrada conectada al jack Line-In de la tarjeta.

Todos estos registros tienen las siguientes características:

3 bits por canal, consiguiendo 8 niveles de volumen para cada uno de los canales izquierdo y derecho.

Valores de 0 a 7 correspondiéndose a una ganancia de -46 dB a 0 dB en saltos aproximadamente de 4 dB.

El valor por defecto varía según el canal. Para los canales Voice, Master y MIDI es 4 (-11 dB) y para los canales CD y Line, es 0 (-46 dB)

EL MIXER DE LA SOUND BLASTER 16 y AWE 32: EL CHIP CT1745

CARACTERÍSTICAS:

Las características del CT1745 superan en mucho a las del CT1345:

Control de Volumen:

Permite 32 niveles de volumen independientes para canales izquierdo y derecho del Master, Voice, MIDI, CD y Line-In.

El micrófono es mono y tiene 32 niveles de volumen

El PC Speaker es mono y tiene 4 niveles de volumen.

Selección de los canales de salida:

La salida final a través del Line-Out es la suma de los canales Voice, MIDI, CD, Line-In, Micrófono y PC Speaker. En el CT1345, el único modo de dejar de oír uno de los canales era reducir el volumen de ese canal a cero. En el CT1745 tres de las entradas (Micrófono, CD y Line-In) pueden desactivarse mediante *switches.*

Selección de los canales de entrada:

En contraposición con el CT1345, es CT1745 permite la grabación simultánea a través del micrófono, CD, Line-In y MIDI (cada una con el volumen deseado). Al igual que en el caso de salida, se puede desactivar un canal de entrada mediante *switches.*

Otra característica es que permite dirigir el canal izquierdo y/o derecho de cualquier fuente de entrada al canal izquierdo y/o derecho del mixer de entrada, de manera que se puede redireccionar el canal izquierdo del Line-In (por ejemplo) al canal derecho del mixer, y el Derecho del Line-In al izquierdo del mixer. Con esto se conseguiría un cambio de los canales izquierdo y derecho a través del hardware. También podríamos dirigir el canal izquierdo de la fuente de entrada a los canales izquierdo y derecho del mixer, y el canal derecho de la fuente de entrada a ambos del mixer, con lo que se conseguiría pasar de estéreo a mono.

AGC (Automatic Gain Control) para el micrófono:

La entrada proveniente del micrófono pasa a través de un AGC que amplifica la señal de entrada automáticamente hasta alcanzar un nivel razonable. Esto ayuda a amplificar menos los sonidos fuertes, y a amplificar mucho mas los débiles.

Control de Ganancia:

Tiene un control independiente de ganancia de 4 niveles para los canales izquierdo y derecho de entrada y de salida.

Control de Tono:

Control independiente de 16 niveles para los canales izquierdo y derecho de agudos y graves. El CT1745 tiene filtros dinámicos, por lo que no necesita control de filtros como el CT1345. Tampoco tiene el *switch que habilitaba el estéreo. Otra diferencia es que en este caso, el volumen del micrófono afecta a la amplitud de la señal tanto cuando se graba como cuando se reproduce.*

REGISTROS DEL MIXER DE LA SOUND BLASTER 16 Y AWE 32

Ahora pasaremos a la explicación de cada uno de los registros del mixer. Para una rápida visión mirar la tabla 2.

Registro 0x00 (Reset)

Escribiendo cualquier valor en este registro, el resto de registros serán restaurados a sus valores por omisión.

Registros 0x04, 0x0A, 0x22, 0x28 y 0x2E (Compatibilidad con el CT1345)

Estos registros sólo están por compatibilidad con el CT1345. Los valores por omisión y las ganancias son diferentes, y se aprovechan mas bits de los que se aprovechaban en el CT1345. De todas maneras es recomendable no utilizar éstos, y en contra utilizar los propios del nuevo chip, ya que se dispone de más niveles de volumen.

Los registros 0x04, 0x22, 0x26, 0x28 y 0x2E, utilizan 4 bits por canal, consiguiendo 16 niveles de volumen. Valores de 0 a 15 se corresponden con ganancias de -60 dB a 0 dB con un salto de 4 dB por valor.

El valor por omisión para los registros 0x04, 0x22 y 0x26 es 12: -12 dB y para los registros 0x28 y 0x2E es 0: -60 dB.

El registro 0x0A utiliza 3 bits, consiguiendo 8 niveles. Los niveles van de 0 a 7 y corresponden a ganancias de -42 dB a 0 dB con pasos de 6 dB. El valor por omisión es 0: -42 dB.

Registros 0x30/0x31 (Volumen de Master Izquierdo/Derecho)

Registros 0x32/0x33 (Volumen del canal Voice Izquierdo/Derecho)

Registros 0x34/0x35 (Volumen del canal MIDI Izquierdo/Derecho)

Registros 0x36/0x37 (Volumen del CD Izquierdo/Derecho)

Registros 0x38/0x39 (Volumen del canal Line Izquierdo/Derecho)

Registro 0x3A (Volumen del micrófono)

Todos estos registros tienen las siguientes características:

5 bits por canal, consiguiendo 32 niveles de volumen para cada uno de los canales izquierdo y derecho.

Índice	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	Reset del mixer							
0x04	Voice izquierdo				Voice derecho			
0x0A						Micrófono		
0x22	Master izquierdo				Master derecho			
0x26	MIDI izquierdo				MIDI derecho			
0x28	CD izquierdo				CD derecho			
0x2E	Line-In izquierdo				Line-In derecho			
0x30	Master izquierdo							
0x31	Master derecho							
0x32	Voice izquierdo							
0x33	Voice derecho							
0x34	MIDI izquierdo							
0x35	MIDI derecho							
0x36	CD izquierdo							
0x37	CD derecho							
0x38	Line-In izquierdo							

Valores de 0 a 31 se corresponden a una ganancia de -62 dB a 0 dB en saltos de 2 dB.

El valor por defecto varía según el canal. Para los canales Voice, Master y MIDI es 24 (-14 dB) y para los canales CD, Line y micrófono es 0 (-62 dB)

Registro 0x3B (Volumen del PC Speaker)

Utiliza 2 bits y por lo tanto tiene 4 niveles.

Los valores de 0 a 3 se corresponden a ganancias de -18 dB a 0 dB en pasos de 6 dB.

El valor por omisión es 0: -18 dB.

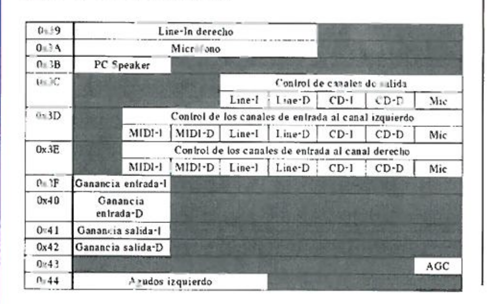
Registro 0x3C (Switches de salida)

Habilita la salida de los canales que tengan el bit activado.

El valor por omisión es: Todos activados.

Registro 0x3D (Switches de entrada del canal izquierdo)

Selecciona las fuentes de entrada que tendrá el canal izquierdo del mixer. Estarán seleccionadas aquellas que tengan el bit activado.



0x39	Line-In derecho							
0x3A	Micrófono							
0x3B	PC Speaker							
0x3C				Control de canales de salida				
				Line-I	Line-D	CD-I	CD-D	Mic
0x3D		Control de los canales de entrada al canal izquierdo						
		MIDI-I	MIDI-D	Line-I	Line-D	CD-I	CD-D	Mic
0x3E		Control de los canales de entrada al canal derecho						
		MIDI-I	MIDI-D	Line-I	Line-D	CD-I	CD-D	Mic
0x3F	Ganancia entrada-I							
0x40	Ganancia entrada-D							
0x41	Ganancia salida-I							
0x42	Ganancia salida-D							
0x43								AGC
0x44	Agudos izquierdo							

Por omisión están seleccionados los canales izquierdos de Line, CD y el micrófono.

Registro 0x3E (Switches de entrada del canal derecho)

Selecciona las fuentes de entrada que tendrá el canal derecho del mixer. Estarán seleccionadas aquellas que tengan el bit activado.

Por omisión están seleccionados los canales derechos de Line, CD y el micrófono.

Cuando se hace una grabación en mono, hay que tener en cuenta que las muestras serán digitalizadas sólo del canal izquierdo. Por lo tanto, si se procede a una grabación de una fuente de entrada estéreo, primero hay que activar los *switches izquierdo y derecho del registro 0x3D de esa fuente ya que si no hacemos esto, se digitalizaría sólo el canal izquierdo, perdiendo de esta manera toda la información del canal derecho.*

Registros 0x3F/0x40 (Ganancia de entrada Izquierdo/Derecho)

Registros 0x41/0x42 (Ganancia de salida Izquierdo/Derecho)

2 bits por canal, consiguiendo 4 niveles.

Valores de 0 a 3 se corresponden a ganancias de 0 a 18 dB en saltos de 6 dB.

El valor por omisión es 0: 0 dB.

Registro 0x43, bit 0 (AGC del micrófono)

0: AGC activado (Valor por omisión)

1: AGC desactivado. La ganancia queda fijada a 20 dB.

Registros 0x44/0x45 (Agudos Izquierdo/Derecho)

Registros 0x46/0x47 (Graves Izquierdo/Derecho)

4 bits por canal, por lo tanto se consiguen 16 niveles.

Valores de 0 a 7: -14 dB a 0 dB, pasos de 2 dB.

Valores de 8 a 15: 0 dB a 14 dB, pasos de 2 dB.

Valor por omisión: 8 (0 dB).

EL PROGRAMA DE EJEMPLO

El programa de ejemplo suministrado esta entrega permite cambiar el mixer de la Sound Blaster. No es un programa útil, ya que estas tarjetas ya traen un programa que lo hace, pero con esto se ampliarán las rutinas que teníamos hechas hasta ahora. Tampoco se ha cuidado para nada la estética del programa.

Se han incorporado nuevos procedimientos y funciones al módulo SB.PAS que se encargan de leer y escribir en el mixer. Estas rutinas proceden de diferente manera según poseamos una SB Pro o una SB 16. Para que las funciones sean mas genéricas, se han hecho de manera que los valores van de 0 a 255, pero luego se convierte este valor a uno dentro el rango apropiado según la tarjeta.

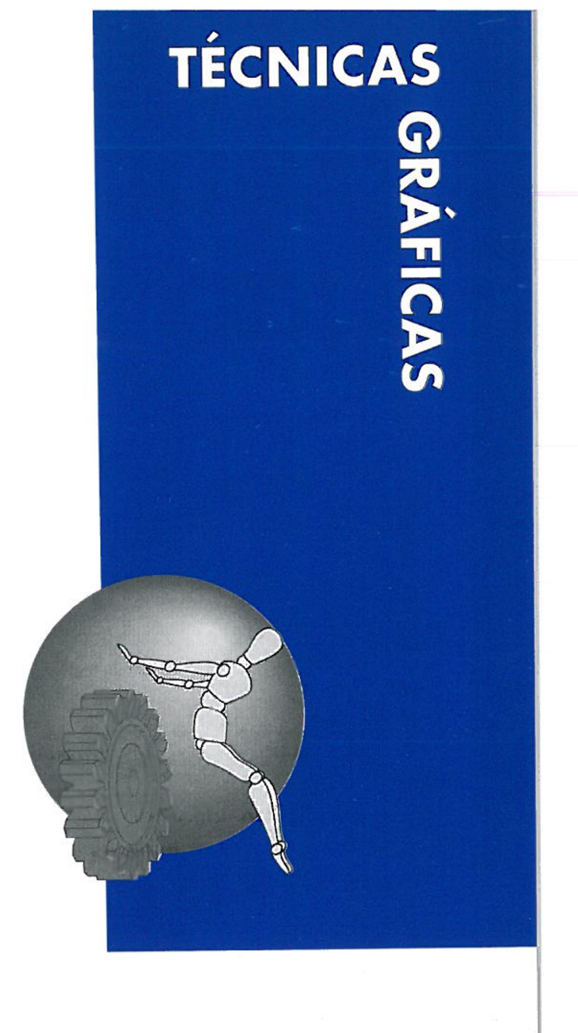
EN LA PRÓXIMA ENTREGA

En este artículo se ha explicado todo lo que quedaba por saber de las tarjetas Sound Blaster. Sería interesante, sin embargo, en algún artículo posterior hacer un resumen de todos los comandos del DSP a modo de referencia rápida, ya que es lo que todo programador utiliza a la hora de hacer un programa.

De todas maneras, para no aburrir a los que no dispongan de una tarjeta de sonido de la familia Sound Blaster, en el próximo número volveremos a nuestro player. Se completará el reproductor mejorando la calidad de sonido, haciendo interpolación de muestras por software, con control de volumen independiente para cada canal, y si el artículo da para más, se podrían llegar a implementar algunos efectos que se han dejado de lado hasta ahora.

RAY TRACING (II) LA ILUMINACIÓN

Alvaro Silgado



En el capítulo anterior se explicó en qué consiste la técnica Ray Tracing para obtener imágenes de síntesis. Se introdujeron unos conceptos básicos de geometría, necesarios para entender esta técnica. Se conocieron las clases elementales que iban a entrar en juego y, por último, se mostró la estructura típica de un programa de Ray Tracing.

Después de tanta historia, lo más que el lector habrá conseguido del programa que se adjuntó serán imágenes "planas", sin volumen. Sí, los objetos se pueden cambiar de sitio, se puede mover la cámara y se puede jugar a cambiar los colores o diseñar patrones divertidos, pero nada más. Y, para ser sinceros, el acabado deja mucho que desear. Pero, si los objetos con los que se está trabando tienen volumen y se encuentran en un mundo tridimensional, ¿por qué aparecen representados como simples "manchas" de colores? Muy sencillo: porque no hay iluminación.

En este capitulo se va a hacer una introducción al concepto de iluminación. Podrá comprobar como las escenas ganan en volumen, expresión y calidad, siendo posible obtener imágenes bastantes vistosas. También se mostrará cómo se hace la intersección con dos de los objetos más utilizados: el plano y el polígono. Además, se va a hacer un replanteamiento del código del programa, haciéndolo realmente orientado a objetos, más sencillo y mantenible.

CAMBIOS, PERO A MEJOR

Antes de continuar con temas propios del Ray Tracing hay que hacer un alto en el camino para fijarse en aspectos concernientes a la programación. El que haya echado un ojo al código del programa adjuntado en el capítulo anterior habrá podido observar que estaba escrito en C++, con los elementos implementados como clases. Pero era una orientación a objetos "entre comillas".

Sí, había encapsulamiento de algunos datos y métodos, había polimorfismo (métodos virtuales), pero la herencia un poco relativa. Por ejemplo, los objetos creados estaban declarados como descendientes de la clase objeto. Pero el constructor de cada objeto iniciaba campos heredados de su clase antecesora, lo cual es absurdo. Además, había objetos, como los rayos, que claramente podían ser declarados como descendientes de otra clase. El motivo de todas estas "chapucillas" fue la claridad del programa. Interesaba más que se entendieran los conceptos clave de cada clase y cómo funcionaban que su diseño.

En esta versión del programa se ha dado un pequeño giro en la estructura del mismo. El objetivo es hacerlo realmente orientado a objetos, ahora que el lector está más o menos familiarizado con todos los elementos que intervienen. Todas las modificaciones están documentadas en el propio código del programa. Que nadie se asuste, que no han sido muchas.

Los cambios más destacables son:

- La clase rayo es descendiente de la clase vector (un rayo es un vector con un punto de origen).
- La clase cámara es descendiente de la clase rayo (una cámara es un rayo, porque tiene un origen y un vector de dirección, además de otros elementos).

En un mundo sin luz ni sombras no hay sensación de volumen. Las imágenes obtenidas a partir de ahora van a ser mucho más reales porque estarán iluiminadas.

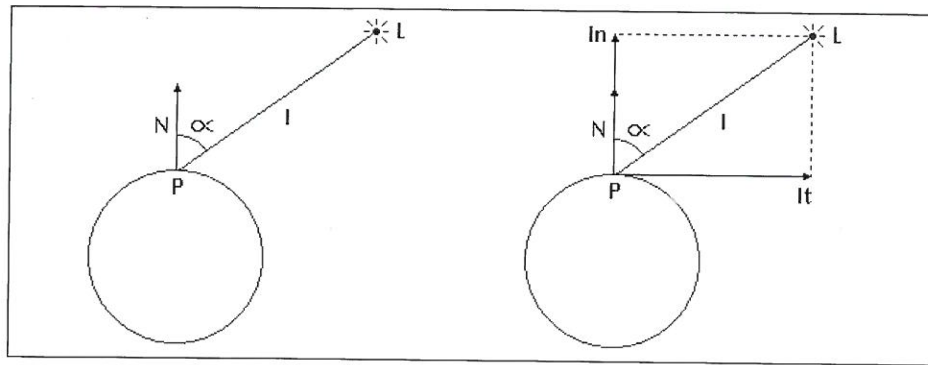


Figura 1. El vector de luz incidente se puede descomponer en un vector normal y uno tangencial a la superficie.

- Los constructores de los objetos sólo inician elementos propios de su clase y no de sus antecesoras.
- Se han programado nuevos constructores de objetos (como en la cámara) que calculan elementos auxiliares que servirán para acelerar el código.
- La clase objeto pasa a ser una clase abstracta, ya que la declaración de sus métodos termina con = 0. De esta forma, no se podrán definir elementos tipo objeto, pero sí descendientes suyos. Además tiene un nuevo campo (si es luminoso o no) y un nuevo método (normal).
- Hay dos nuevos tipos de objeto: el plano y el polígono.
- El objeto mundo tiene un campo nuevo (la luz ambiental) y un método nuevo (iluminar).

Y SE HIZO LA LUZ

Para poder implementar algo tan complejo como la luz, primero es necesario entender cómo se comporta al incidir sobre una superficie. La figura 1 muestra un objeto sobre el cual incide la luz proveniente del punto L. En el punto P, el vector I (que va desde L hasta P) forma un ángulo Alfa con el vector N, que es normal a la superficie en ese punto. Dicho vector I representa la dirección de la energía generada por el punto L y puede ser descompuesto en dos vectores, uno normal (In) y otro tangencial (It). O dicho de otra forma:

In = I * cos (Alfa)
It = I * sen (Alfa)

De esta manera, se puede observar que la cantidad de energía que realmente incide sobre el punto P es proporcional al vector In, produciéndose degradados de color según varíe Alfa. A esto se le conoce con el nombre de luz difusa. Pero, ¿cómo se obtienen colores a partir de esta fórmula? Es sencillo. La luz puede descomponerse en sus tres colores primarios Rojo (r), Verde (g) y Azul (b). Por lo tanto, si el objeto tiene en el punto P un color o coeficiente de luz difusa (Pr, Pg, Pb) y el punto L emite una luz (Lr, Lg, Lb), la luz que emitirá el punto P (la luz que se ve en dicho punto) es:

r = Pr * Ir * cos (Alfa)
g = Pg * Ig * cos (Alfa)
b = Pb * Ib * cos (Alfa)

Hay otro factor que interviene en la iluminación: la distancia. La intensidad de la luz proveniente de una lámpara que recibe un objeto es inversamente proporcional a la distancia que les separa al cuadrado. Es decir, cuanto más lejos está una lámpara de un objeto, menor es la intensidad de luz que éste recibe. Aunque en la realidad la intensidad de la luz disminuye con la distancia al cuadrado, en simulación se obtienen mejores resultados con la distancia sin elevar al cuadrado. Además, es más rápido.

Por lo tanto, habrá que calcular la distancia que separa la lámpara del punto de intersección y dividir:

r = Pr * Ir * cos (Alfa) / D
g = Pg * Ig * cos (Alfa) / D
b = Pb * Ib * cos (Alfa) / D

Esto es en el caso de que sólo haya una luz. Si hay varias luces, se hará la suma de la energía aportada por cada una de ellas, resultando:

r = Pr * (I1r * cos (Alfa1) / D1 + ... + Iir * cos (Alfai) / Di)
g = Pg * (I1g * cos (Alfa1) / D1 + ... + Iig * cos (Alfai) / Di)
b = Pb * (I1b * cos (Alfa1) / D1 + ... + Iib * cos (Alfai) / Di)

Además, hay definida una luz ambiental, que es una cantidad de energía constante para todos los elementos de la escena. La luz ambiental se produce al rebotar la luz en los átomos de la atmósfera y difundirse uniformemente. Por lo tanto, si la luz ambiental viene definida por (Ar, Ag, Ab), la expresión queda:

r = Ar + Pr * (I1r * cos (Alfa1) / D1 + ... + Iir * cos (Alfai) / Di)
g = Ag + Pg * (I1g * cos (Alfa1) / D1 + ... + Iig * cos (Alfai) / Di)
b = Ab + Pb * (I1b * cos (Alfa1) / D1 + ... + Iib * cos (Alfai) / Di)

QUIEN DICE LUZ, DICE SOMBRA

Hay un factor importantísimo que se ha pasado por alto: la sombra. Si un objeto se interpone entre el punto que produce la luz y el punto que se está estudiando, dicho punto estará en sombra y, por lo tanto, no se tendrá en cuenta la aportación de energía de dicha luz.

Para estudiar si un punto está iluminado por una lámpara, se genera un rayo que parte de dicho punto y va hacia ella. Supóngase que la lámpara se encuentra situada a una distancia D del punto. Se van haciendo intersecciones con los objetos definidos. Si no

```
(r, g, b) es el color que se está calculando.
(Ar, Ag, Ab) es la luz ambiental.
(Pr, Pg, Pb) es el color del punto P.
(Iir, Iig, Iib) es el color de la luz i-ésima.
Ii es el vector que une el punto P y la luz i-ésima.
N es el vector normal a la superficie en el punto P.

r = Ar
g = Ag
b = Ab
para i = 1 hasta num_luces
   si la luz i-ésima ilumina al punto P entonces
      cos_alfa = (Ii escalar N) / distancia
      r = r + Pr * Iir * cos_alfa
      g = g + Pg * Iig * cos_alfa
      b = b + Pb * Iib * cos_alfa
   fin del si
fin del para

devolver (r, g, b)
```

Figura 2. Algoritmo de cálculo de luz.

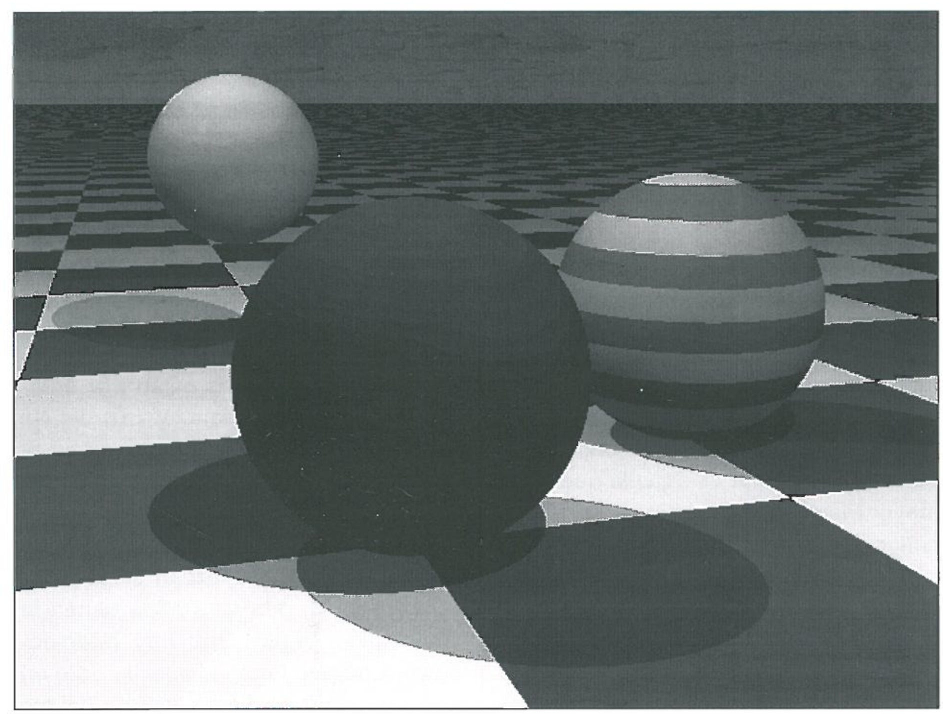


Figura 3. Las imágenes generadas ya tienen luces y sombras.

hay ningún objeto que haga intersección con el rayo, el punto está iluminado por la lámpara. Si algún objeto hace intersección y el punto de intersección esta a una distancia D' del punto, éste estará en sombra si 0 < D' < D. Es decir, si la intersección se produce a medio camino entre el punto de estudio y la lámpara.

La figura 2 muestra el algoritmo que calcula la luz para un punto dado.

LOS VECTORES NORMALES

Para poder implementar el algoritmo anteriormente descrito, sólo hace obtener el vector normal (o perpendicular) a una superficie en un punto dado. Este vector tiene que ser calculado por el propio objeto, ya que solamente él conoce su geometría. Hay que recordar que en todo momento se trabaja con instancias de la clase genérica objeto, de las cuales sólo se sabe cómo generarlas.

En el capítulo anterior se decía que la forma de crear un nuevo tipo de objeto es definir su función de intersección con un rayo. A esto, hay que añadirle otra función que devolverá el vector normal a la superficie, dado un punto perteneciente al objeto.

Para el objeto esfera creado en el capítulo anterior, el vector normal en un punto P es el vector que va desde el centro de la esfera hasta dicho punto. Hay que recordar que todos los vectores directores tienen que estar normalizados. Por lo tanto, habrá que llamar a la función normalizar pasándole el vector obtenido. En este caso en concreto, se puede normalizar el vector dividiendo todas sus componente por el radio de la esfera, siempre y cuando se esté seguro que el punto P va a pertenecer siempre a la superficie de ésta.

El objeto suelo siempre tendrá el mismo vector normal: apuntando hacia arriba, es decir, (0, 1, 0).

Una vez obtenido el vector normal, el cálculo del coseno de Alfa es sencillo, ya que el producto escalar de dos vectores normalizados devuelve precisamente eso, el coseno del ángulo que forman.

UN NUEVO OBJETO: EL PLANO

Capítulo a capítulo, se va a ir estudiando toda una colección de objetos que pueden ser añadidos a las imágenes que se estén diseñando. Uno de los más utilizados es el plano. Hay que tener en cuenta que un plano es infinito. Por lo tanto, el uso principal que se le va a dar es para definir paredes, techos, etc., de tal forma que sea delimitado por otros objetos. La ecuación de un plano es:

A * x + B * y + C * z + D = 0

Todos los puntos que cumplan esta condición, pertenecen al plano definido. Va a ser útil saber que en esta ecuación, (A, B, C) definen el vector normal del plano, mientras que D es la distancia de éste al origen. Por lo tanto, la función que devuelve el vector normal es muy fácil: el vector normal siempre es (A, B, C) para cualquier punto. Se ha aprovechado el campo pos de la clase genérica objeto para guardar dicho vector y se ha añadido un campo a la clase plano para guardar D.

La intersección de un plano con un rayo no es muy complicada. Hay que recordar que la ecuación paramétrica que define un rayo es:

R(t) = Ro + Rd * t, con t > 0.

Ro es el punto de origen, definido como (xo, yo, zo). Rd es el vector director, definido como (xd, yd, zd). Sustituyendo en la expresión del rayo, se obtiene:

x = xo + xd * t
y = yo + yd * t
z = zo + zd * t

Sustituyendo estas expresiones en la ecuación del plano, queda:

A * (xo + xd * t) + B * (yo + yd * t) + C * (zo + zd * t) + D = 0

Como siempre, la incógnita es "t", que es la distancia del punto de intersección al comienzo del rayo. Despejando la incógnita:

t = - (A * xo + B * yo + C * zo + D) / (A * xd + B * yd + C * zd)

Si se expresa de forma vectorial, la expresión final queda de la siguiente forma:

t = - (N | Ro + D) / (N | Rd)

siendo N el vector normal del plano y "|" el producto escalar de dos vectores. Primero se calculará N | Rd. Si el resultado es 0, el rayo es paralelo o está incluido en el plano, en cuyo caso no se tendrá en cuenta la intersección.

EL POLÍGONO, EL DESCENDIENTE DEL PLANO

El otro objeto que se va a estudiar es el polígono. Un polígono es el conjunto ordenado de todos los vértices que forman una superficie cerrada incluida en un plano. Por lo tanto, todos los puntos que componen el polígono son coplanares, es decir, todos pertenecen al mismo plano. Se puede decir que un polígono está incluido en un plano, o dicho de otra forma, un polígono es un plano delimitado.

Es por este motivo por el que se ha programado la clase polígono como descendiente de la clase plano. Además, se ha añadido un array de puntos y una variable que indica cuántos puntos hay. De esta forma, la función que devuelve el vector normal es la misma para un plano que para un polígono, y la de intersección de un polígono se basará en la de intersección de un plano.

Se ha programado un constructor que, dado el array de puntos, calcula el vector normal del plano y su distancia al origen. Para ello, busca tres puntos del polígono consecutivos y diferentes entre sí. Con ellos se construyen dos vectores de tal forma que el producto vectorial normalizado de ambos resulta ser el vector normal del plano. La distancia es muy sencilla de calcular, basta con sustituir en la ecuación del plano las componentes del vector normal obtenido y despejar D.

Algo más compleja resulta la función de intersección. Lo primero que hace es llamar a la función de intersección del plano del que desciende. Si no hay intersección positiva con el plano, tampoco la habrá con el polígono. En el caso de que sí exista intersección, se calculará el punto en el que se produce ésta. De lo que se trata es de saber si el punto está dentro o fuera del polígono.

DENTRO O FUERA

Lo primero que se va a hacer para ver si un punto está dentro de un polígono es hacer una proyección de ambos en dos dimensiones, ignorando una de las tres componentes de cada punto. Los ejes de coordenadas de la proyección se denominan U y V. La componente que se va a ignorar es aquella que tiene mayor magnitud en el vector normal del polígono, ya que será la que menos información proporcione de su geometría. A esta componente se le llama dominante, y es precalculada al generar el polígono. Si el punto proyectado está dentro del polígono proyectado, el punto en el espacio estará dentro del polígono en el espacio, y viceversa.

A continuación, se resta a cada componente de cada punto del polígono proyectado la correspondiente componente del punto proyectado, produciendo una traslación del polígono de tal forma que el punto de estudio ahora es el origen de coordenadas.

Para saber si el punto está dentro o fuera del polígono, se calcula el número de intersecciones del polígono proyectado con la parte positiva del eje U. Si el número de intersecciones es par, el punto está fuera del polígono. Si es impar, está dentro.

Gracias a la traslación que se ha hecho, calcular el número de intersecciones no es muy complicado. Se irán estudiando los vértices del polígono de dos en dos. Si ambos están por encima ($V > 0$) o por debajo ($V < 0$) del eje U, no pueden hacer intersección. Si uno está encima y otro debajo, pero ambos tienen la U negativa, tampoco hay intersección, ya que sólo interesan aquellas que se produzcan en la parte positiva de U, pero si ambos tienen la U positiva, seguro que hay intersección y no hace falta calcularla. Sólo en el caso de que uno esté encima del eje U, el otro debajo y sólo uno de ellos tenga la U positiva, se calculará la intersección para ver si es positiva. Si el primer punto está definido como (Ua, Va) y el segundo (Ub, Vb), esta intersección viene dada por la fórmula:

$$U = Ua - Va * (Ub - Ua) / (Vb - Va)$$

Si $U > 0$, hay intersección en la parte positiva de U. Ya sólo resta contar el número de intersecciones y ver si es par o impar.

Si un vértice cae justo en el eje, va a dar lo mismo, ya que se considerará que un punto está debajo del eje U si $V < 0$, y se considerará que está encima si $V >= 0$. De esta forma, un punto nunca puede estar en el eje.

ESTO YA PARECE ALGO

Los resultados obtenidos al aplicar la luz difusa son bastante espectaculares. No hay más que comparar una imagen generada con la versión anterior del programa con la misma imagen generada con la nueva versión. Y seguirán siendo cada vez más reales, según se introduzcan nuevos aspectos como la reflexión y la refracción. Además, los nuevos objetos permiten definir escenas más complejas, así que... rienda suelta a su imaginación.

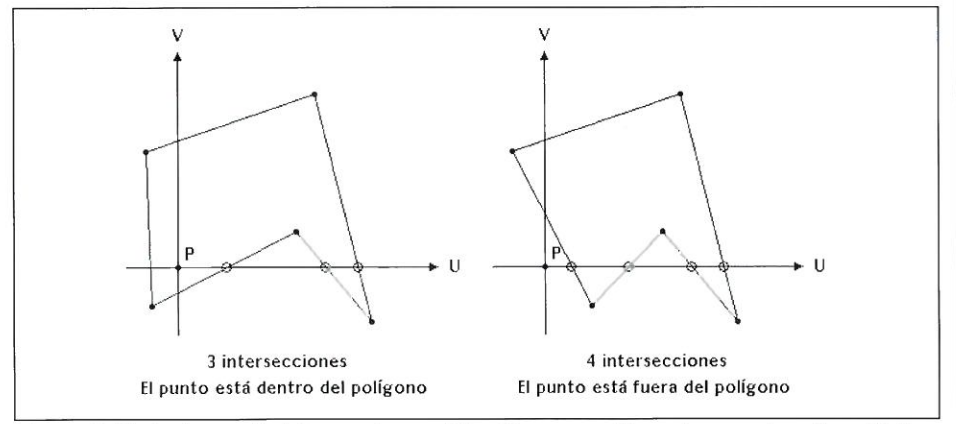
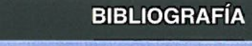


Figura 4. Si el número de intersecciones del polígono con U+ es impar, el punto está dentro.



BIBLIOGRAFÍA

- "An Introduction to Ray Tracing". Editado por Andrew S. Glassner. Editorial Academic Press.
- "Fractal programming and Ray Tracing with C++". Roger T. Stevens. Editorial M&T Publishing, Inc.
- "Computer Graphics. Principles and practice". Foley, van Dam, Freiner y Hughes. Editorial Addison Wesley.
- "Models of light reflection for computer synthesized pictures". Blinn, J.F. Editado en Siggraph 1.977.

PROGRAMACIÓN DEL GDI

Jorge R. Regidor

En la programación para Windows, hasta el hecho de mostrar un texto en pantalla implica una operación gráfica. Como todos sabemos, Windows es un entorno gráfico y como tal hay que hacer un uso exhaustivo de estas capacidades. Para ello, dentro del API existen un sin fin de funciones, macros y tipos encargados de facilitarnos el trabajo en la programación gráfica. Además, este conjunto de funciones son totalmente independientes del hardware de cada equipo, lo que implica no tener que preocuparse de manera alguna del tipo de la tarjeta gráfica, fabricante o revisión, ya que de eso ya se preocupa Windows o el propio fabricante suministrando los drivers oportunos.

De lo que si se debe uno preocupar es de las capacidades del hardware instalado. Por ejemplo, en un PC se puede tener instalado una flamante SVGA con 2Mb de memoria, una resolución de 1280x1024 pixels y 65535 colores, y en otro PC una tarjeta VGA de 512K con una resolución de 640x480 pixels y 16 colores. Por tanto, el programa a de ser capaz de detectar las capacidades gráficas del sistema, que estarán íntimamente ligadas a la tarjeta de vídeo del equipo.

Hay otra característica importantísima del GDI que hay que tratar. GDI también es independiente del medio a mostrar los resultados de una operación, así podremos mostrar un gráfico sobre pantalla o sobre una impresora o plotter con las mismas funciones, pero seleccionando un dispositivo u otro. La única restricción, parte de las capacidades gráficas de cada equipo.

CONTEXTO DE DISPOSITIVOS

Los contextos de dispositivos o Device Context(DC) son unas estructuras que recogen un conjunto de objetos gráficos, atributos asociados a estos y modos gráficos que afectan al dispositivo de salida de las operaciones gráficas. De tal forma que cualquier salida a pantalla, impresora u otro dispositivo gráfico se realizará utilizando estos contextos de dispositivo.

Pero, ¿qué son los objetos gráficos? Los objetos gráficos incluyen las plumas(Pens) para dibujar líneas, brochas(Brushes) para dibujar fondos y rellenar figuras, los mapas de bits(Bitmaps) para mostrar imágenes, una paleta(Palette) para indicar los colores disponibles por el dispositivo, las fuentes(Fonts),... La Figura 1 muestra los objetos gráficos y sus atributos asociados.

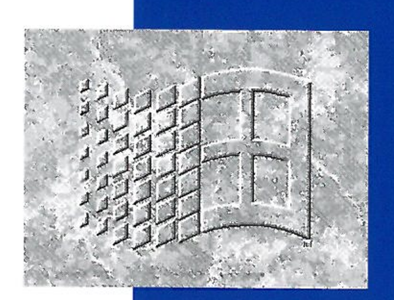
MODOS GRÁFICOS

Existen cinco modos gráficos bajo Windows, que permiten especificar como los colores serán mezclados, como aparecerán los resultados en el dispositivo de salida o como serán escalados en la salida. Estos modos están en el contexto de dispositivos y son los siguientes:

* *Background:* Define como los colores de fondo serán mezclados con los colores actuales de la ventana.

* *Drawing:* Define como los colores de primer plano serán mezclados con los colores actuales de la ventana.

* *Mapping:* Define como será el sistema de coordenadas para mostrar la salida en una ventana, la pantalla o la impresora.



Toda la potencia y facilidad a la hora de realizar tareas gráficas desde Windows es facilitada por una de las tres librerías de enlace dinámico que forman el núcleo Windows, por supuesto, hablamos de GDI (Graphics Device Interface). Esta librería nos permite de manera independiente del hardware, realizar todas las tareas gráficas que deseemos.

* *Polygon-fill:* Define el método por el cual se van a rellenar los polígonos.

* *Stretching:* Define el método a utilizar para reducir o ampliar los gráficos en la salida.

Para establecer o observar las valores de los diferentes modos existen diferentes funciones descritas en la Figura 2.

Tipos de contextos de dispositivo

Existen cuatro tipos de contextos de dispositivo en la programación para Windows, cada uno de ellos con una labor muy específica. Veámoslos más detalladamente.

Contexto de dispositivo de Display

Dentro de este tipo de contextos, a su vez, existen tres tipos, class, common y private. Class y private son utilizados en aquellas aplicaciones que hagan un uso intensivo de las salidas gráficas, y common es utilizado en aplicaciones con uso infrecuente de los gráficos.

Class Device Contexts

Este tipo de dispositivos de contexto están soportados solamente por compatibilidad con anteriores versiones de Windows, por lo tanto hay que rehusar su utilización.

Common Device Contexts

Este tipo de contextos de dispositivo de display, mantienen una cache con el gestor de ventanas de Windows, por tanto cada vez que se requiera una salida gráfica en pantalla, debemos obtener un contexto de dispositivo disponible, que Windows inicializa con los valores por defecto (Véase Figura 3). Una vez realizadas todas las operaciones se debe liberar el contexto de dispositivo, ya que existe un número limitado de estos para todas las aplicaciones Windows.

En la programación gráfica con este tipo de dispositivos, hay que resaltar tres puntos importantes. El primero, la obtención de un dispositivo de contexto con GetDC o BeginPaint; segundo el establecimiento de los valores de los objetos gráficos y su utilización; y tercero la liberación del contexto mediante ReleaseDC o EndPaint. La utilización de GetDC/ReleaseDC frente a BeginPaint/EndPaint, parte de la inclusión del código o no dentro del mensaje WM_PAINT. si se pinta dentro de este mensaje se debe utilizar BeginPaint/EndPaint, de lo contrario hay que utilizar GetDC/ReleaseDC. El Listado 1, ilustra el proceso descrito dentro de un mensaje WM_PAINT.

```
case WM_PAINT:
{
  HDC hDC;
  PAINTSTRUCT psPaint;

  /*** Obtención de un contexto de dispositivo de dis-
play ***/

  hDC = BeginPaint(hWnd, &psPaint);

  /*** ... Utilización de los objetos gráficos ...***/

  /*** Liberación del contexto ***/

  ReleaseDC(hWnd, hDC);
}
```

Listado 1. Código para utilizar contextos de dispositivo common.

Private Device Contexts

Este tipo de contextos de dispositivo mantienen una contexto común a toda la clase de ventanas de ese tipo. Al no tener una cache intermedia, cuando se utilizan estos contextos no deben ser liberados.

En la programación de este tipo de contextos, solamente hay que obtener el handle al contexto una vez, ya que todos los cambios realizados sobre este tipo de contextos se mantienen hasta la liberación de esa clase de ventanas. Para especificar que una clase de ventanas tiene un contexto de dispositivo privado, hay que indicar el estilo CS_OWNDC al registrarla. El contexto de dispositivo será liberado al destruir la última ventana de dicha clase. En el Listado 2, se muestra un ejemplo de programación con este tipo de contextos de dispositivo.

```
HDC hDC; // Variable global para el handle al con-
texto de dispositivo

BOOL
IniciaAplicacion(HINSTANCE hInstance)
{
  WNDCLASS wndMiWindow;

  /*** Registrar la clase con el estilo CS_OWNDC
***/

  wndMiWindow.style =      CS_VREDRAW      I
CS_HREDRAW I CS_OWNDC;
          wndMiWindow.lpfnWndProc          =
(WNDPROC)wndProcMiWindow;
  wndMiWindow.cbClsExtra = 0;
  wndMiWindow.cbWndExtra = 0;
  wndMiWindow.hInstance = hInstance;
      wndMiWindow.hIcon  =  LoadIcon(NULL,
IDI_APPLICATION);
    wndMiWindow.hCursor  =  LoadCursor(NULL,
IDC_ARROW);
          wndMiWindow.hbrBackground        =
GetStockObject(WHITE_BRUSH);
  wndMiWindow.lpszMenuName = NULL;
  wndMiWindow.lpszClassName = "MiWindow";

  return (RegisterClass(&wndMiWindow));
}
.
.
.
case WM_CREATE
{
  hDC = GetDC(hWnd):
  return 0;
}
case WM_... // Mensaje donde se requiere una sali-
da gráfica
{
  // Utilización de los objetos gráficos sobre hDC

  /*** No hay que llamar a ReleaseDC ***/
}
case WM_PAINT:
{
  PAINTSTRUCT psPaint;

  BeginPaint(hWnd, &psPaint);

  // Utilización de los objetos gráficos sobre hDC

  /*** No hay que llamar a EndPaint ***/
}
```

Listado 2. Programación utilizando contextos de dispositivo private.

Contexto de dispositivo de impresora

Sólo existe un tipo de contexto de dispositivo para todos los tipos de impresoras, ya sean Láser, chorro de tinta o matricial.

Para imprimir cualquier tipo de objeto gráfico se debe abrir el contexto de dispositivo de impresora mediante la función CreateDC. Una vez que haya sido utilizada la impresora se debe eliminar el contexto de dispositivo con DeleteDC. Nunca se debe utilizar ReleaseDC para liberar el contexto.

Objeto gráfico	Atributos asociados
Bitmap	Tamaño, en bytes; dimensiones, en pixels; formato de color, esquema de compresión...
Brush	Estilo, color, patrón y origen.
Palette	Colores y tamaño (número de colores).
Font	Nombre del tipo, ancho, alto, grosor, conjunto de caracteres...
Pen	Estilo, ancho y color.

Figura 1. Objetos gráficos y sus atributos.

Contexto de dispositivo de memoria o compatible

Este tipo de contextos de dispositivo, mantienen una copia en memoria con todos los valores de los objetos gráficos de un contexto de dispositivo de display. Sirven para realizar operaciones con bitmaps, tales como mostrar, escalar o crear animaciones mezclando varios de ellos. Para crear un contexto de este tipo hay que utilizar la función CreateCompatibleDC pasándole el handle del contexto con el cual se quiere ser compatible. Para terminar su uso, hay que llamar a la función DeleteDC.

Contexto de dispositivo de información

Son un tipo especial de contextos de dispositivo utilizados para obtener información de los valores de sus objetos. Como la información se puede obtener sin tener que crear todas las estructuras necesarias, la utilización de este tipo de contextos es mucho más rápida. Para destruirlo, después de su uso, se llama a la función DeleteDC.

Utilización de objetos gráficos

Como ya se ha reseñado anteriormente, cada contexto de dispositivo tiene una serie de objetos asociados con unos valores por defecto que contienen el modo en el cual se van a realizar las operaciones gráficas tales como dibujar una línea, un circulo, rellenar una figura, mostrar una imagen o incluso mostrar un texto.

Existen varias funciones para obtener y establecer los valores de los distintos objetos gráficos. Para obtener las propiedades de los objetos por defecto de Windows se utiliza la función GetStockObject que tiene el siguiente formato.

HGDIOBJ GetStockObject(fnObject);

donde,

fnObject especifica el tipo de objeto que deseamos elegir. Entre los posible valores están:

BLACK_BRUSH: Brocha de color negro.
DGRAY_BRUSH: Brocha de color gris oscuro.
GRAY_BRUSH: Brocha de color gris.
HOLLOW_BRUSH: Brocha transparente.
LTGRAY_BRUSH: Brocha de color gris claro.
NULL_BRUSH: Sin Brocha. Transparente.
WHITE_BRUSH: Brocha blanca.
BLACK_PEN: Pluma negra.
NULL_PEN: Sin Pluma.
WHITE_PEN: Pluma blanca.
ANSI_FIXED_FONT: Fuente de ancho fijo. Juego de caracteres ANSI.
ANSI_VAR_FONT: Fuente de ancho variable. Juego de caracteres ANSI.
DEVICE_DEFAULT_FONT: Fuente por defecto del dispositivo.
OEM_FIXED_FONT: Fuente de ancho fijo. Juego de caracteres OEM.
SYSTEM_FONT: Fuente de ancho variable usada por Windows por defecto para menús y diálogos.
SYSTEM_FIXED_FONT: Fuente de ancho fijo usada por Windows por defecto para menús y diálogos.

Esta función se combina con SelectObject para seleccionar un objeto gráfico determinado dentro de contexto de dispositivo. El siguiente listado muestra el modo de hacerlo.

```
case WM_...
{
 HBRUSH hBr;
 HBRUSH hBrOld;
 HDC hDC;

 hDC = GetDC(hWnd);
 hBr = (HBRUSH)GetStockObject(GRAY_BRUSH);
 hBrOld = SelectObject(hDC, hbr);

 // Operaciones con la brocha seleccionada

 SelectObject(hDC, hbrOld);
 DeleteObject(hBr);
 ReleaseDC(hWnd, hDC);
}
```

En el listado puede verse como se obtiene el handle (HBRUSH) de la brocha de color gris y se selecciona dicha brocha en el contexto de dispositivo de display. Después de esto se pueden realizar las operaciones con esta bro-

Atributo de DC	Valor por defecto
Color de fondo	Blanco.
Tipo de fondo	Opaco.
Color de la brocha	Blanco.
Color de la pluma	Negro.
Color del texto	Negro.
Fuente	Fuente del sistema, proporcional.
Región de recorte	Área de trabajo.
Origen DC	(0,0) vertice superior izquierdo del área de trabajo.
Posición de la pluma	(0,0).
Posición de la brocha	(0,0).

Figura 3. Valores por defecto de los contextos de dispositivo.

Modo gráfico	Establecer	Observar
Background	SetBkMode	GetBkMode
Drawing	SetROP2	GetROP2
Mapping	SetMapMode	GetMapMode
Polygon-Fill	SetPolyFillMode	GetPolyFillMode
Stretching	SetStretchBltMode	GetStretchBltMode

Figura 2. Modos gráficos en Windows

cha sobre el contexto. Cuando se termine su utilización, se debe restablecer la brocha anterior, eliminar la brocha creada y liberar el contexto de dispositivo. Programa ejemplo. Capacidades del sistema.

En el programa ejemplo de este artículo, se ha mostrado el uso de otra función muy importante en el GDI. Dicha función se encarga de obtener las capacidades gráficas del sistema actual, de tal forma que en tiempo de ejecución nos permita conocer las limitaciones del mismo. Esto sirve por ejemplo, para impedir que se intente visualizar un bitmap de 256 colores sobre una tarjeta gráfica que sólo soporte 16.

El formato de dicha función es el siguiente:

```
int GetDeviceCaps(hdc, iCapability)
```

donde,

En la programación para Windows, hasta el hecho de mostrar un texto en pantalla implica una operación gráfica

hdc es el handle del contexto de dispositivo del cual queremos saber sus capacidades.

iCapability es un identificador que indica la capacidad sobre la cual se desea información. La Figura 4 muestra algunas de estos identificadores de capacidades y su significado.

Identificador	Descripción
DRIVERVERSION	Versión del dispositivo.
TECHNOLOGY	Tecnólogia del dispositivo. Los posibles valores son: DT_PLOTTER, DT_RASDISPLAY, DT_RASPRINTER, DT_RASCAMERA, DT_CHARSTREAM, DT_METAFILE, DT_DISPFILE.
HORZSIZE	Tamaño horizontal, en milímetros.
VERTSIZE	Tamaño vertical, en milímetros.
HORZRES	Resolución horizontal, en pixels.
VERTRES	Resolución vertical, en pixels.
LOGPIXELSY	Númro de pixels por pulgada en altura.
LOGPIXELSX	Númro de pixels por pulgada en anchura.
BITSPIXEL	Número de bits por pixel.
PLANES	Número de planos de color.
NUMBRUSHES	Número de brochas del dispositivo.
NUMPENS	Número de plumas del dispositivo.
NUMFONTS	Número de fuentes del dispositivo.
NUMCOLORS	Número de colores del dispositivo.

Figura 4. Identificadores de capacidades de un dispositivo.

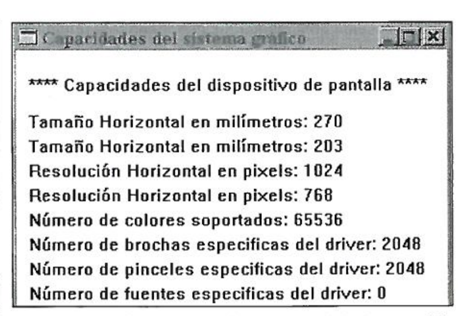


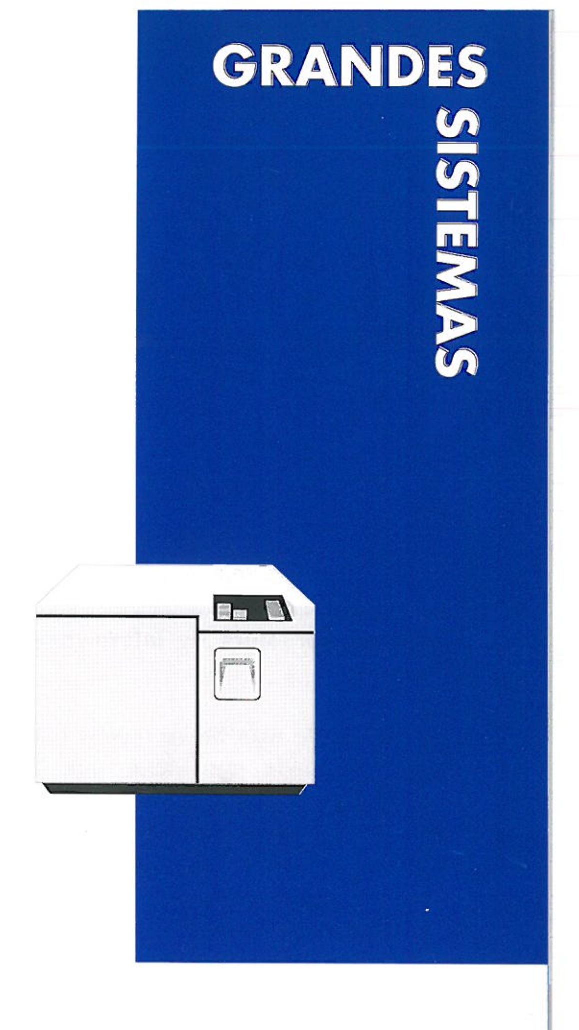
Figura 5. Listado de las capacidades gráficas. Programa ejemplo.

En el programa se muestran algunas de las capacidades gráficas del display utilizando esta función y los resultados se muestran en formato de texto (Véase Figura 5). En el siguiente artículo, este programa será ampliado utilizando algunos de los objetos gráficos como plumas, fuentes, brochas y bitmaps.

CONCLUSIÓN

En este articulo se ha pretendido sentar la base de la programación del GDI, para posteriormente realizar el uso de sus contextos de dispositivo y objetos gráficos, sabiendo el porqué de cada acción realizada. De no tener claro que es un contexto de dispositivo, el motivo de crearlos o liberarlos, que es un objeto gráfico o como se seleccionan y liberan; se puede caer en la frivolidad de programar el GDI y no saber que se hace. Y por experiencia propia, no hay nada más sencillo para que una aplicación no funcione correctamente que utilizar indebidamente los contextos de dispositivo y sus objetos.

BASES DE DATOS (I)



José María Peco

El autor de este artículo no pretende profundizar en conceptos tan amplios como los que aquí se van a enunciar, sólo intenta matizar términos y conceptos que son de uso común; y, de paso, hacer que este artículo sirva de introducción a los temas que se trataran los próximos meses en esta sección: Fundamentos de las Bases de Datos ADABAS y DB2.

UN POCO DE HISTORIA

Aunque nunca se haya estudiado este tema, todo el mundo ha oído hablar del término BASE DE DATOS, y más o menos tiene una idea de su significado.

Para Charles W. Bachman, diseñador del IDS (Integrated Data Storage) , primer Sistema de Gestión de Base de Datos, abreviadamente SGDB, el mundo de las Bases de Datos nació con las grabaciones en cinta magnética de los años 50.

Pero la verdad es que al principio sólo había "datos", o conjuntos de caracteres encadenados que reproducían las características de los objetos o hechos que describían.

Estos datos, que podían formar grupos repetitivos, se organizaron en "ficheros" o "archivos", los cuales recogieron como herencia la organización de los ficheros de ficha de cartulina tan extendidos en esa época.

Estos ficheros, base de la llamada "Informática de Gestión", contenían la información de la realidad cotidiana, asociaciones, correspondencias, uniones entre elementos, etc., generando con ello una red en la que los elementos que la configuraban no podían tratarse de forma independiente.

El siguiente paso, por tanto, fue la aparición de los "ficheros consolidados", cuyos elementos se informaban unos a otros cuando ocurría un suceso, es decir, cada vez que se modificaba o añadía un registro, se actualizaban los punteros que le unían al resto de elementos. Esto permitía mantener una cierta coherencia en los datos, pero el coste de proceso que esto suponía llegaba a dar al sistema una pesadez de tratamiento a veces insoportable, siendo esta una característica importante de los sistemas de gestión de los años 60-70

Llegados a este punto, hay que destacar el hecho de que paralelamente, grandes empresas, organismos estatales (no olvidemos que el Gobierno de los Estados Unidos es el primer consumidor de informática del mundo) y paraestatales empezaron a interesarse por el concepto "banco de datos", como acumulación sistemática de informaciones sobre un campo de estudio particular. Este concepto pretendía tratar la información como si se tratara de un banco, cuya moneda fuera precisamente la información. Esta organizacion, llamada "cooperativa de Información" a veces también recibe el nombre de Base de Datos, lo cual no es coherente ni se ajusta a la definición que sigue.

Definición de BD

En la ultima década, en las grandes instalaciones de España se ha producido un gran auge de los Sistemas Gestores de Bases de Datos. Dos son los que, atendiendo al tipo y a los sistemas sobre los que corren, han destacado respecto al resto: ADABAS y DB2.

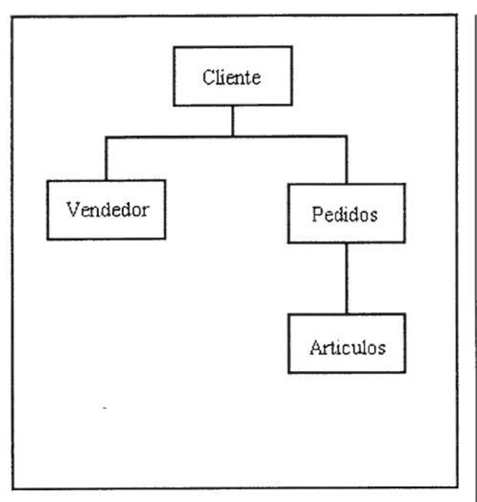


Figura 1. Estructura de Información Jerarquica.

Después de esta breve reseña histórica, se puede considerar que la Base de Datos nace de la convergencia de necesidades específicas de las empresas y de las necesidades de organizar gigantescas agrupaciones de datos a escala profesional o nacional.

La definición que se da a continuación del significado de la Base de Datos se basa principalmente en criterios técnicos mas que en criterios de utilización, y pretende recoger todos los conceptos y características de una base de datos.

Una base de datos es un conjunto (a menudo voluminoso, pero no necesariamente) de datos almacenados en soportes externos en los que se permite el acceso directo. Estos datos son utilizables por los programas de aplicación (batch u on-line) a fin de realizar altas, bajas, modificaciones y búsquedas, para lo cual disponen de procedimientos automáticos de búsqueda. Dichos datos están integrados, es decir, cada dato se encuentra almacenado una sola vez, tanto en su contenido propio como en las relaciones con otros datos de la misma base, relaciones que pretenden ser el fiel reflejo de las existentes en la realidad concreta que se desea representar.

¿ Por qué las Bases de Datos necesitan un sistema gestor ?

Cuando se habla del registro de un fichero se esta hablando de un conjunto de información que tiene una estructura determinada.

Para poder leer la información contenida en un fichero, es necesario realizar las siguientes operaciones:

- *Asignar el fichero* al entorno del programa que le quiere usar (es lo que se entiende por ALOCAR el fichero).

- *Abrir el fichero,* para lo cual debe ejecutarse la rutina del sistema operativo que realiza esa función.

- *Leer el registro,* es decir, obtener una copia de la información que buscamos, desde el soporte en el que se encuentra (disco por ejemplo) sobre el buffer o área de memoria que el programa ha reservado para él.

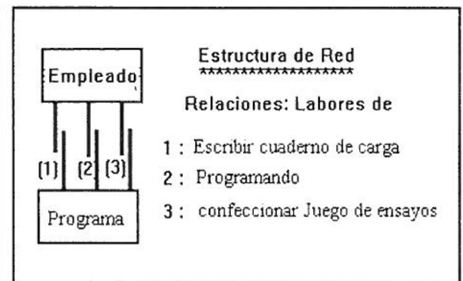


Figura 3.

- *Hacer uso* de dicha información.

De esta forma, se ha traído al entorno del programa toda la información contenida en un registro. Pero, para poder hacer uso de esta información, en tiempo de codificación se ha tenido que redefinir el buffer sobre el que se va a depositar esta información. Es decir, se ha definido la plantilla que permitirá separar la lista de caracteres y números que componen el registro, en los distintos campos que dan significado a esa información.

Por otra parte, si se desea leer ese mismo fichero desde otro programa, de nuevo hay que incluir esa plantilla u otra en el nuevo programa, para que nos permita definir cada una de las posiciones del registro, de modo que, por ejemplo, las 15 primeras posiciones constituyen el nombre, que las 10 siguientes constituyen el sueldo, que las 8 siguientes constituyen la fecha de nacimiento, etc.

Esto presenta una serie de inconvenientes, como por ejemplo,

- todos los programadores tienen acceso a toda la información del registro, pues éste se lee entero.

- Un mismo campo puede definirse y tener nombres distintos en distintos programas.

Para solventar, entre otros, estos inconvenientes, en los sistemas multitarea aparecieron los Sistemas Gestores de Bases de Datos.

¿Qué es un Sistema Gestor de Bases de Datos o SGDB ?

Un SGBD realmente es un conjunto de programas independientes de la

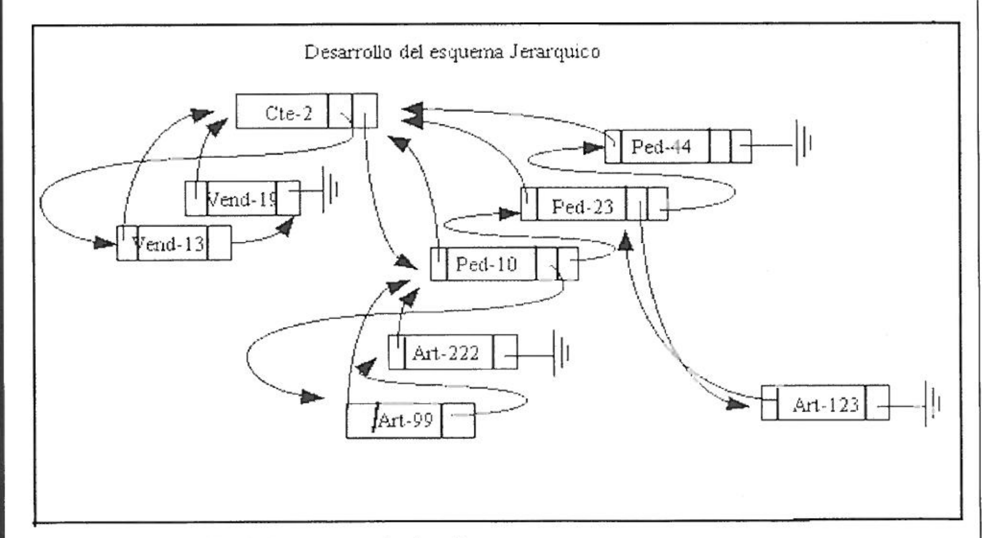
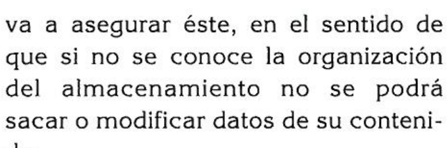


Figura 2. Desarrollo del esquema Jerárquico.



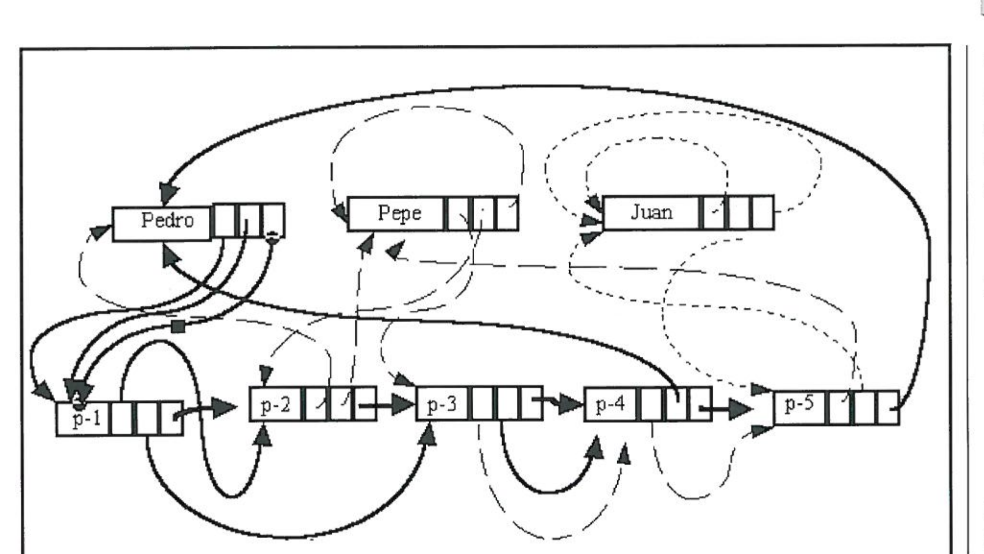


Figura 4. Desarrollo de la red.

aplicación que se esta ejecutando, y que junto con una serie de ficheros auxiliares, facilitan la gestión de recuperación de información y de grabación de la misma en los soportes magnéticos que la contendrán de forma permanente.

Cuando la información se encuentra en una base de datos, el programador no necesita saber si un determinado campo ocupa las 15 primeras posiciones o las 15 últimas del registro, sólo sabe que el campo se llama NOMBRE y que forma parte del fichero de PERSONAL, y que tiene una longitud de 15 bytes, por ejemplo.

informacion en el buffer habilitado al efecto.

Al hacerlo de este modo, se resuelven los dos problemas que se planteaban en el apartado anterior, pues:

- sólo se recuperan los campos que se especifiquen siempre y cuando se esté autorizado para su manejo (ya que al ser el SGBD un programa que se interpone entre el programa de aplicación y el soporte físico de los datos, se le puede especificar determinados niveles de autorización para devolver determinada información).

El SGBD determina como suyos todos los ficheros de la base de datos

Por otra parte, el SGBD determina como suyos todos los ficheros que van a componer la base de datos, por lo que ningún otro programa de aplicacion podrá alocar dichos ficheros. Esto presenta al menos la ventaja de que esos ficheros se encuentran permanentemente alocados desde que se "arranca" la Base de datos, por lo que no hay que alocarles cuando se desea hacer uso de un fichero, sólo hay que llamar al SGBD diciéndole que se desea recuperar los campos que se especifican de un registro determinado de tal o cual fichero, y ya él se encarga de devolver esa

- Todos los programas usarán el mismo nombre para referirse al mismo campo, ya que ese nombre es el que conoce el SGDB.

TIPOS DE SISTEMAS GESTORES DE BASES DE DATOS

Como ya hemos dicho, una Base de datos debe, ser considerada, no sólo por los soportes de la información, si no que además debe considerarse el Sistema Gestor de la Base de Datos o SGDB, es decir, el software que, al tiempo que sirve para independizar al usuario del almacenamiento, va a asegurar éste, en el sentido de que si no se conoce la organización del almacenamiento no se podrá sacar o modificar datos de su contenido.

Fundamentalmente son 4 los sistemas gestores :

- Jerárquicos
- En red.
- Basados en listas Invertidas
- Relacionales

Los sistemas Jerárquicos implementan estructuras basadas en un modelo lógico donde cada entidad se descompone de modo progresivo en otras entidades cada vez mas elementales. El SET o registro lógico esta formado por un segmento raíz, del que pueden colgar varios hijos, de los que a su vez pueden colgar otros, etc. Cada uno de estos segmentos dependientes, están unidos entre si mediante distintos tipos de punteros que facilitan la recuperación de la información.

La figura 1 muestra un modelo de este tipo de estructura, mientras que la figura 2 muestra la implementación de una ocurrencia concreta.

Las bases mas representativas de este tipo fueron IMS de IBM y SYSTEM-2000

Los sistemas en RED sustentan un modelo de datos que admite que un miembro tenga varios propietarios, pudiendo estos pertenecer a la misma o a varias clases de entidad propietario. El método usado para implementar este modelo se basa en punteros, los cuales materializan las relaciones entre los distintos elementos de las entidades.

La figura 3 muestra un modelo de este tipo de estructura, y la figura 4 muestra la implementación de una realidad concreta.

En este tipo de estructuras , se deben diferenciar las que seguían las normas CODASYL, tales como IDS, IDS II (Honnewell Bull), IDMS (IBM), y DMS (Univac)

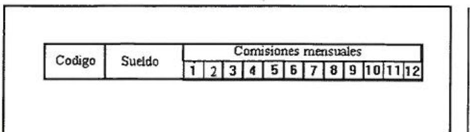


Figura 5. Relación sin normalizar.

Sin Normas Codasyl: TOTAL (IBM, Honnewell, Univac) (sólo admitía dos niveles)

Los sistemas basados en listas Invertidas:

El principio de este modelo consiste en asignar al registro lógico un identificador único, y usar dicho numero en la confección de las listas que permitirán recuperar la información en base a los campos claves que se hayan definido.

Este es el modelo adoptado para la gestión de la base de datos ADABAS. Los sistemas Relacionales:

Este tipo de base de datos se fundamenta en el concepto matemático de RELACION.

Así, una relación n-aria R definida en los conjuntos C1, C2, C3, ... Cn no necesariamente disjuntos, es un subconjunto del producto cartesiano de esos n conjuntos.

A cada elemento de una relación binaria se le conoce con el nombre de "par ordenado" o "Tupla", pues está formado por un elemento de C1 y otro de C2 y en este orden.

No obstante, y a pesar del enfoque matemático de este planteamiento, no se debe caer en la trampa de pensar que los conceptos vertidos en las BDRs no tienen nada que ver con los conceptos informáticos, ya que como

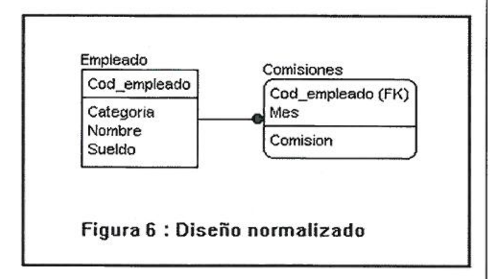


Figura 6 : Diseño normalizado

Figura 6.

se vera en próximos artículos, es posible traducirlos a algo que nos es tan familiar como las tablas de doble entrada.

Ejemplos de bases de Datos que implementen este modelo, si hacemos caso a la propaganda del fabricante, son prácticamente todas las que hoy en día se encuentran en el mercado, tales como Oracle, Informix, etc, pero, en grandes sistemas la que se destaca es DB2 (de IBM).

CARACTERÍSTICAS GENERALES DE LAS BD.

Como características generales cabe destacar las siguientes:

- Las Bd. suelen ser voluminosas, lo cual no quiere decir que porque un fichero de datos sea muy voluminoso, sea una base de datos.

- Las Bases de datos deben tener en cuenta su capacidad de crecimiento en un futuro.

- La integración de los datos se encuentra dentro de la Base de datos, pues cada aplicación no usa sus ficheros, sino que todos usan los mismos ficheros.

- Las Bases de datos, tal y como se desprende de la definición dada, han de tener acceso directo a los datos.

- Actualmente se entiende por base de datos, no sólo el almacenamiento de las bases de datos, sino que esta incluido el sistema de gestión de bases de datos

Por último, cabe destacar que aunque la definición dada anteriormente, especifica que los datos están almacenados una sola vez, es decir, que no hay redundancia, eso no es totalmente cierto en ninguno de los sistemas comentados, si bien, esta redundancia cada sistema tiende a que sea mínima y que este controlada por el propio sistema.

Formas Normales

Al diseñar el modelo de datos en el que se basará una aplicacion, se deben tomar una serie de precauciones a fin de evitar problemas a la hora de realizar altas, bajas y/o modificaciones de las tuplas. Las formas normales pretenden mantener la estructura lógica de los datos en una forma normal y limpia.

Se dice que una relación esta sin normalizar cuando existen tuplas en los que algún atributo es múltiple. (figura-5)

Primera forma normal (1FN): El modelo se dice que esta normalizado hasta la primera forma normal (1NF en Inglés) cuando ninguna entidad del modelo contiene atributos múltiples. (Figura-6)

Segunda Forma Normal (2FN): Esta forma se cumple cuando, además de cumplirse la Primera forma normal, todo atributo no primario, depende de la clave concatenada en su totalidad. Es decir, los atributos no clave dependen de la totalidad de la clave concatenada.

Tercera Forma Normal (3FN): Esta forma se cumple cuando, además de cumplirse la Segunda forma normal, cada atributo no primario depende de una única clave primaria. Es decir, los elementos no clave dependen de una única clave.

Cuarta Forma Normal (4FN): Esta forma introduce el concepto de dependencia multivalor o multivaluada (DMV o MVD).

Quinta forma normalizada (5FN): Esta forma se basa en el concepto de dependencia de asociación o dependencia JOIN.

Integridad referencial

Este concepto, básico en los SGBD, establece que una tupla de una relación que haga referencia a otra relación, debe referirse a una tupla existente de esta otra relación. Este principio, asegura por lo tanto, la congruencia entre las tuplas de las distintas relaciones.

EL SONIDO EN LINUX

David Aparicio



Aun sin ser imprescindible, el sonido es una de las facetas que un sistema operativo debe afrontar para cumplir las exigencias de la multimedia actual. Hoy realizaremos una primera exploración sobre este tema en Linux.

Con el abaratamiento y la popularización de las tarjetas de sonido, hoy día es frecuente disponer de una de ellas en nuestro propio PC. Desde el clásico sonido de 8 bits, hasta la perfección digital del compact-disc, desde el sonido FM hasta las moderna generación por tabla de síntesis, hay un amplio abanico de hardware al alcance de todos los bolsillos. Linux, como cualquier sistema operativo de hoy, debe sacar partido a este material para proporcionar soporte a las aplicaciones multimedia, así como a los juegos, cotidianos en el ordenador de nuestra casa.

UN POCO DE HISTORIA

En el nacimiento del proyecto Linux, se entendió a este sistema como exclusivo para desarrollo. Hoy día, podemos comprobar que intérpretes, compiladores y herramientas de programación siguen siendo las más abundantes. Sin embargo, parte del esfuerzo se concentra ahora en fomentar la creación de aplicaciones de usuario, verdadero soporte de un sistema.

La existencia de un API (Aplication Programming Interface: conjunto estandarizado de llamadas al sistema) que aísle a las aplicaciones de los posibles cambios que se introducen en el núcleo de Linux se hace imprescindible. Si en las llamadas al propio sistema operativo se sigue el estándar Posix, acercándolo al mundo compatible Unix, en el entorno de ventanas se utiliza el estándar X-Windows, del que incluso el estándar MS-Windows toma como raíz. En gráficos a pantalla completa, la librería SuperVga trata de unir a Linux con FreeBSD, otro clónico Unix.

Nos queda un aspecto importante, si no fundamental, como es el manejo del sonido. El finés Hannu Savolainen es el autor de VoxWare, el conjunto de llamadas que intenta estandarizar el tratamiento del sonido tanto en Linux como en otros sistemas compatibles Unix. Hannu se percató del problema al que se enfrentaba cualquier programador para asimilar la información necesaria y para manejar tan extenso abanico de hardware, y la barrera de la poca portabilidad para migrar aplicaciones de sonido de un entorno a otro.

Con cada revisión del núcleo de Linux se proporciona el núcleo de VoxWare, que ofrece un interfaz único para cualquier hardware que sea soportado en el núcleo de sonido. Por ejemplo, el control de volumen (mezclador) se ve desde la aplicación como un sistema que admite valores entre cero y cien para cada "mando", independientemente del número real de niveles que admita el hardware. La aplicación sabe que, por ejemplo, cincuenta es un valor medio, y no se preocupa de las operaciones efectivas que VoxWare realice en la tarjeta. De este modo, será mas fácil para el desarrollador hacer aplicaciones que se concentren en la potencia y vistosidad, abstrayéndole del problema de interaccionar con el hardware. Adicionalmente, una aplicación diseñada para otro Unix será fácilmente transportable a Linux, fomentando el desarrollo entre plataformas, y aumentando el abanico de aplicaciones disponibles. Podemos ver un esquema de este entorno en la figura 1.

UNA PRIMERA TOMA DE CONTACTO

Antes de pasar a conocer el aspecto del sistema de sonido, repasemos la pecu-

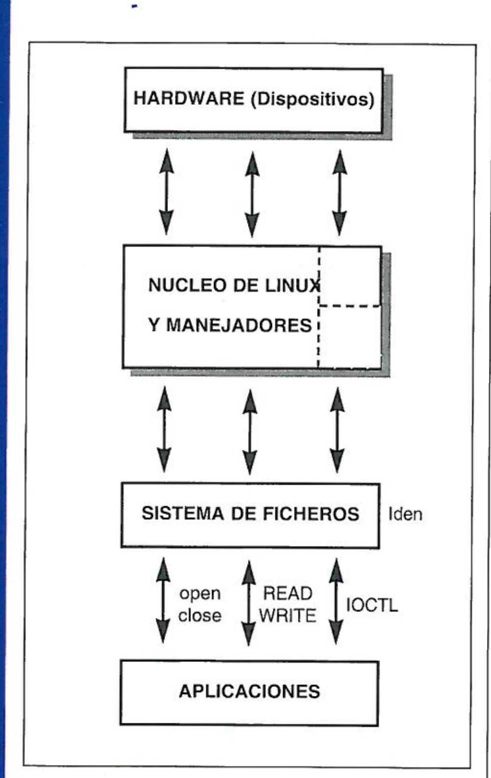


Figura 1: "El sonido en Linux".

liaridad de ciertos directorios en la estructura de ficheros de Linux.

Uno de ellos es /proc, que nos permite comunicarnos con el núcleo, principalmente para obtener información de procesos ejecutándose y dispositivos instalados. El número y tipo de ficheros en esta estructura no pueden ser creados ni borrados directamente, sino que dependen de la versión y funcionalidad del núcleo que tengamos ejecutándose.

Otro directorio especial es /dev, que nos permite interactuar con los dispositivos reconocidos por el núcleo. Debido al proceso de instalación, seguramente ya se habrá familiarizado con los que representan al sistema de almacenamiento: hda1, fd0, sdb ...

Las pantallas de texto también tienen sus dispositivos asociados (tty), así como las líneas serie (ttyS y cua). Aunque los nombres pueden diferir algo, el modo de funcionamiento que vamos a comentar a continuación es común en todas las versiones Unix.

Los nombres de ficheros en este directorio representan dispositivos físicos. Debido a que su naturaleza es, por tanto, diferente a la de los ficheros comunes, hay un comando especial para crearlos: mknod. Con este comando debemos indicar si el dispositivo es de tipo carácter o bloque. El primero representa pantallas, impresoras, lineas serie, y todo tipo de dispositivos con los que se puede dialogar carácter a carácter. Los dispositivos de bloque, donde la velocidad de transferencia es muy alta, pero no tanto su método de búsqueda, toman un cluster (bloque, o grupo de caracteres, de tamaño prefijado) como unidad mínima de diálogo. Además de por su tipo, los dispositivos se identifican dentro del núcleo con un número mayor y otro menor. Teclee

```
cat /proc/devices
```

Observará un listado de los periféricos detectados en su máquina, con unos números asociados a ellos. Este es el denominado "número mayor". El núcleo asocia, por convención, un identificador a cada manejador de dispositivo, e identifica de forma única a cada tipo de nuestro sistema. El "número menor", por contra, distingue los diferentes dispositivos del mismo tipo, es decir, controlados por un mismo manejador. Por ejemplo, si tenemos varias líneas serie, todas tendrán el mismo número mayor, y número menor diferente. Teclee

```
ls -l /dev/ttyS?
```

Observará que el número mayor es 4 (lo que coincide con el listado de /proc/devices), y el número menor comienza en 64 para el primer puerto serie. Por ejemplo, el primer puerto serie se habría creado con el siguiente comando:

```
mknod /dev/ttyS0 c 4 64
```

en el que se indica el numero mayor y menor, que el dispositivo es de tipo carácter, y el nombre del fichero asociado a crear.

Cuando abrimos uno de estos ficheros especiales, mediante las llamadas open o fopen, internamente se inicializa el dispositivo, y queda preparado para atender a nuestro programa. Podemos indicar el modo en el que deseamos esta apertura. Por ejemplo si indicamos que queremos realizarla en modo exclusivo, podemos impedir efectos nefastos, como en el manejo de /dev/lp0 (impresora), donde a mitad de imprimir una hoja se nos podría "colar" el texto lanzado por otro usuario desde otro programa que también ha abierto el dispositivo.

Las operaciones normales se realizan mediante operaciones de lectura y escritura Por ejemplo, la captación de caracteres del teclado, o la escritura en pantalla, se realiza con llamadas read o write sobre el dispositivo /dev/console (que previamente se ha debido abrir con la llamada correspondiente).

Las operaciones especiales, como cambiar la velocidad de una línea serie, pedir información sobre el estado del dispositivo, y otras, se realiza con la llamada ioctl. Dicha llamada requiere dos parámetros básicamente: un número de función y una estructura donde enviar o recoger los datos que se necesitan. El conjunto de funciones estandarizadas que podemos invocar sobre cada dispositivo es el API antes mencionado, para dicho dispositivo.

Por último, el final de las operaciones y la liberación del periférico se indica con el cierre del fichero asociado.

Una vez conocidos los rudimentos de la interacción con los dispositivos mediante sus ficheros asociados, pasaremos a ver los que nos ofrece VoxWare, a saber:

- /dev/mixer: Este dispositivo permite acceder al mezclador disponible en muchas tarjetas de sonido, controlando el volumen de entrada y salida de cada aspecto de la misma. Sólo admite llamadas ioctl.
- /dev/sndstat: Aquí encontramos un dispositivo que sirve únicamente para comprobar si se ha reconocido correctamente nuestro hardware. Sólo admite llamadas read.
- /dev/dsp: Es el dispositivo principal. Cualquier dato escrito en este dispositivo se reproduce en el dispositivo DAC, PCM o DSP de la tarjeta. Los datos leídos son adquiridos desde el micrófono de la misma. El control del dispositivo que se utiliza en cada momento se selecciona con las llamadas ioctl.
- /dev/audio: Similar al anterior en filosofía, este dispositivo intenta ser

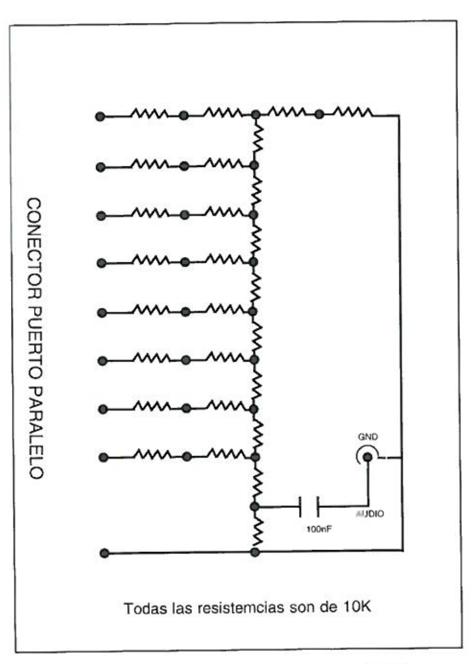


Figura 2: "Un sencillo conversor DAC para puerto paralelo".

compatible con el que se encuentra en las estaciones de trabajo de Sun, para poder acceder al software multimedia disponible para ellas. Utiliza un sistema de compresión logarítmica denominado ley-A, donde las muestras son adquiridas con una resolución de 12 bits, pero se almacenan como 8 bits. Todavía no está completamente implementado en Linux.

- /dev/sequencer: Este periférico utiliza el sintetizador de algunas tarjetas (OPL-3, YM3812) o el sistema de generación por tabla de síntesis de otras. Hasta la versión 3.0 de VoxWare, también se emplea para un acceso rudimentario al puerto MIDI si se encuentra disponible.
- /dev/midi: A partir de la versión 3.0, este dispositivo de tipo carácter permite controlar la cadena MIDI, con el sistema MPU-401 o compatible, en modo inteligente (asíncrono).

Hemos de notar que, si no se interfieren entre sí, por ejemplo por estar configuradas sobre el mismo puerto de entrada/salida, podemos tener varias tarjetas de sonido conectadas en un mismo PC, ya que Linux es capaz de manejarlas todas a la vez. Por ejemplo, si tenemos un mezclador en cada tarjeta, se podrá acceder a ellos mediante /dev/mixer y /dev/mixer1, de forma independiente.

MANEJANDO LOS DISPOSITIVOS

Incluso sin ninguna aplicación específica, podemos ser capaces de observar la capacidad de sonido del sistema. El comando

```
cat /dev/sndstat
```

nos muestra un listado de todos los dispositivos disponibles mediante nuestra tarjeta de sonido, y la configuración detectada. Si este comando nos responde con algo similar a "No such device", bien no disponemos de hardware para sonido, bien no hemos compilado el núcleo con esta opción.
Si ese no es el caso, y dispone de un micrófono, teclee

```
cat /dev/dsp >/tmp/mivoz
```

Hable al micrófono, y cuando acabe, pulse control-C. Teclee ahora

```
cat /tmp/mivoz >/dev/dsp
```

Si todo ha ido bien, estará escuchando lo que ya ha grabado. Ante tal sencillez para adquirir y reproducir con estos simples comandos, se preguntará entonces por la necesidad de tener aplicaciones para controlar el sonido. Bien repita el proceso, esta vez con el dispositivo /dev/audio. Comprobará que el resultado no ha sido tan bueno como anteriormente. Esta es una de las razones por las que se requieren programas especiales. El comando "cat" es adecuado para manejar señales lineales, como las que se manejan con /dev/dsp, pero no es capaz de tratar con la compresión logarítmica de la ley-A.

Otra razón es el control de la calidad de sonido que manejamos. Por defecto, se adquiere y reproduce a 8 bits, y a 8 Khz. Para variar estos parámetros, así como otros aspectos, como el volumen, debemos acudir a aplicaciones específicas.

FORMATOS DE SONIDO EN LINUX

Con la prueba anterior, hemos obtenido el fichero /tmp/mivoz, que contiene únicamente los datos propios del sonido. Nosotros sabemos que se trata de una muestra de sonido de 8 bits/8 Khz porque hemos creado el fichero, pero otra persona, o nosotros dentro de algún tiempo, o una aplicación, necesitaría saber dicha información, incluso para reconocer como una muestra de sonido a dicho fichero. Claramente, necesitamos un formato estándar para nuestros ficheros. Para nuestro pesar, en lugar de un formato, tenemos una gran variedad de ellos, unos más flexibles que otros: VOC (Creative Labs), WAV (Windows wave), AU (formato ley-A, de Sun Sparc), HCOM y AIFF (Apple), SND (Next), CDR (Audio CD), etcétera.

En lugar de contribuir al caos, y proporcionar un nuevo formato, diferente de los anteriores, Linux es capaz de reconocer cualquiera de ellos, a través de un conversor. El programa sox (SOund eXchanger) nos permite pasar de cualquier formato a cualquier otro, y su uso es muy sencillo. Por ejemplo, para el fichero /tmp/mivoz, donde su formato se denomina RAW (del inglés "plano", o sin formato), el comando sería como sigue:

```
sox -V -t raw -r8000 -ub -c1 mivoz
mivoz.wav
```

Indicando que sea explícito con las operaciones (V), el formato de entrada, de 8 Khz, muestras de byte sin signo y un canal. El conversor intenta deducir de la extensión que indiquemos en el fichero todos los datos posibles, para ahorrarnos teclear demasiado. Con ello, acabamos de convertirlo en un fichero con formato compatible Windows. Para aquellos que conozcan este formato, sabrán que sólo es capaz de almacenar ondas de 11, 22 ó 44 Khz, con lo que nuestra muestra de 8 Khz no se ajusta a ninguna de ellas. Por interpolación, el programa de conversión puede insertar muestras adicionales para solucionar este problema. No debe llevarse la impresión de que este programa siempre necesita tantos datos. Por ejemplo, para pasar el fichero WAV a VOC, es tan simple como esto:

```
sox -V mivoz.wav mivoz.voc
```

Facilidades como esta hacen de él un programa imprescindible para utilizar el sonido en Linux. Hemos incluido el ejecutable y la página del manual con el disco de la revista.

LA NATURALEZA DEL SONIDO

En general, todos estos formatos comentados antes tienen que ver con el sonido digitalizado. Para obtenerlo, cada cierta fracción de segundo, un conversor analógico a digital (ADC) "escucha" la línea de entrada, y anota el valor que lee, bien en 8 o 16 bits, en mono o en estéreo, escribiéndolo en memoria. Por ejemplo, en la muestra /tmp/mivoz se han tomado 8 mil muestras por segundo, y grabando en 8 bits, mono. Cuando se reproduce, hay otro conversor digital a analógico (DAC) que toma esas muestras y las deposita en los puertos de salida, con la misma frecuencia y resolución con que se grabaron. Cuanta mayor sean la frecuencia y resolución, más indistinguible del original es el resultado para nuestro oído. Por supuesto todo tiene un precio, pues cuanto mayor fidelidad tenga el sonido grabado, mas espacio ocupará el fichero resultante. Por ejemplo, la máxima calidad (la de compact-disc) es de 44 Khz, a 16 bits y estéreo. Es decir, cada segundo ocupa 44000 x 4 bytes. Una canción de cinco minutos resultaría en poco menos de 53 Mbytes. Dicho de otro modo, un buffer de sonido de 64 Kbytes se agota, a esta velocidad, en algo más de un tercio de segundo. Esto es muy costoso en CPU, como veremos más adelante. El conversor ADC se maneja leyendo, y el DAC escribiendo sobre el dispositivo /dev/dsp.

En el sonido sintetizado, por contra, se emplea un integrado de FM especial, donde se puede generar una onda básica, y modificar ciertos parámetros de la misma para producir sonido. Es el proceso empleado en ordenadores clásicos, como el venerable Commodore 64, y su principal desventaja reside en la dificultad que implica obtener los parámetros adecuados para que el sonido resultante sea realista. La gran ventaja es el poco espacio que se requiere para almacenar un sonido de este tipo.

Para tomar lo mejor de ambas técnicas, las modernas tarjetas de sonido, como la Gravis Ultrasound o la Creative AWE32, emplean lo que se denomina "generación por tabla de onda". Parte del proceso requiere un sonido que ha sido grabado de una fuente real, con un procesado ADC. Esta muestra de sonido se denomina patch. El hardware especial de la tarjeta permite modificar este sonido original, para producir variedad de efectos, y escalas musicales completas. Para este proceso, la tarjeta de sonido debe contar con su propia memoria donde realizar las operaciones. El manejo, carga y descarga de los patches en memoria se realiza mediante un gestor apropiado. En el API de VoxWare hay un embrión para desarrollar uno.

Sox es un programa que permite cambiar el formato de los ficheros de sonido

El formato MOD, que ha sido exhaustivamente tratado en otra sección de esta revista, utiliza una técnica análoga, siendo posible su reproducción en cualquier tarjeta de sonido. En aquellas en las que esta capacidad no está implementada por hardware, se requiere que la CPU de nuestro ordenador realice todos los cálculos necesarios con anterioridad a la propia reproducción. Esto puede llegar a cargar tanto nuestro sistema que no sea capaz de realizar ninguna otra tarea, algo anecdótico en MSDOS, pero nefasto en Linux. Para ver la importancia de esto, en pruebas con un 486DX2/66 y una GUS Max, la carga de reproducción de un fichero MOD alcanza entre un 2% y un 5% de la carga total.

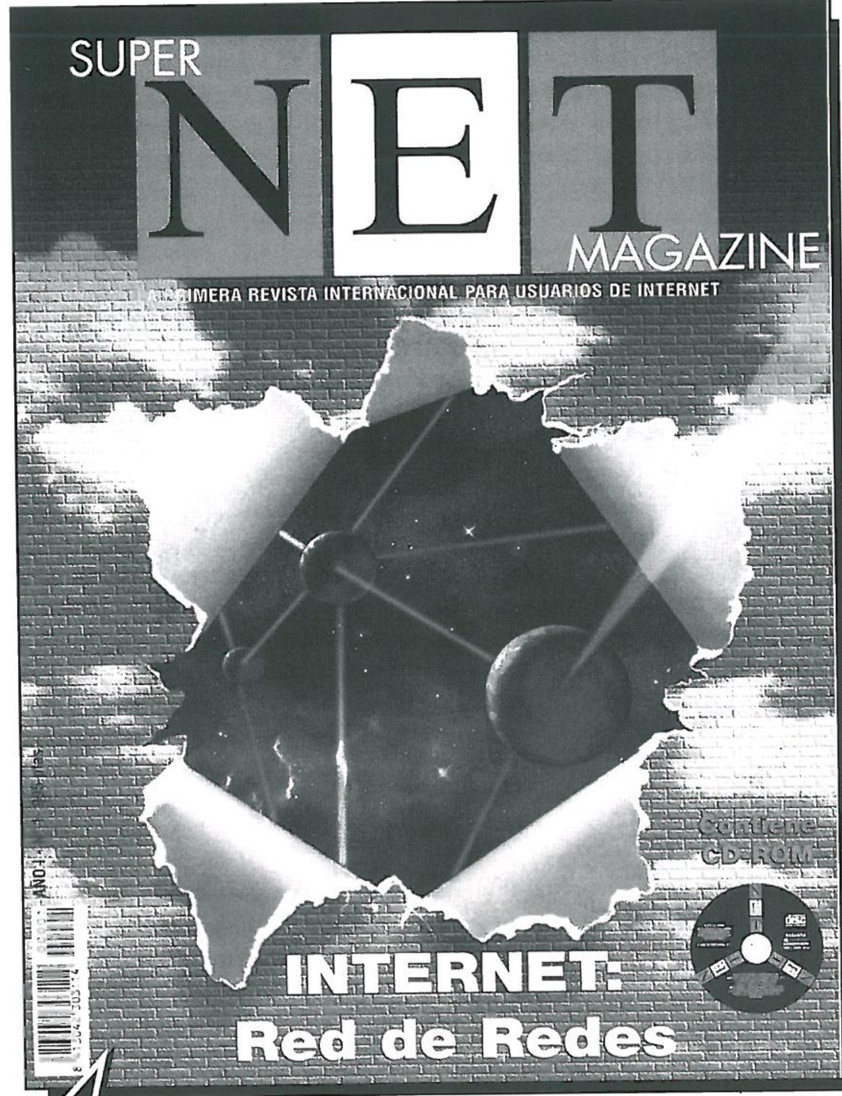
En tarjetas en las que se dispone de un chip DSP, la compresión logarítmica se puede procesar en tiempo real con lo que, con el dispositivo /dev/audio, las prestaciones del sistema se verán beneficiadas. Sin embargo, si no se dispone de esta capacidad, la única forma de saberlo será la parte de CPU que nuestra aplicación consumirá.

Para la reproducción de sonido digital, y si el hardware lo permite, Linux utiliza transferencias DMA. Esto es, suponiendo que hay un buffer de reproducción con memoria contigua, se indica a la tarjeta donde esta el buffer y qué tamaño tiene. Entonces, la CPU puede dedicar su tiempo a otros procesos, mientras la tarjeta de sonido lee los datos de memoria y los reproduce. Hay que tener en cuenta que estamos en un sistema multitarea. Cuando la tarjeta acaba el buffer, lo indica a la CPU mediante una interrupción. Por supuesto, si utilizásemos uno sólo, desde que la CPU se percatase de la señal hasta que programase un nuevo buffer, se producirían unos instantes de silencio inaceptables para la reproducción del sonido. Por ello, se emplean varios bufferes simultáneamente, de modo que mientras la CPU rellena uno y programa su transferencia, la tarjeta de sonido se encuentra reproduciendo otro. Si empleamos búferes suficientemente grandes, la CPU tiene todavía tiempo para atender a otros procesos. Es más, si el sistema estuviese demasiado cargado con, por ejemplo, una compilación, podría darse el caso de que se agotase el buffer de reproducción antes de que el proceso con el programa reproductor consiguiese la atención de la CPU. Si recordamos el ejemplo comentado antes, la reproducción con calidad de un CD consume un buffer de 64Kbytes en un tercio de segundo. Podemos imaginar la carga que debe satisfacer el PC para mantener un sonido de esta calidad.

EL SONIDO, CON COSTE MINIMO

Aunque el coste de una tarjeta de sonido es ya asequible, todavía hay muchos usuarios que no disponen de la misma. Por ello, adjuntamos en la figura 2 un esquema de un reproductor DAC que se conecta a un puerto paralelo, con el que podremos conseguir una calidad de sonido aceptable. Por algo mas de 600 pesetas (y mucha paciencia) el autor de este artículo construyó dicho dispositivo, con mas bien nula experiencia en electrónica. Con 27 resistencias de 10K y cuarto de watio, un condensador de

POR FIN A LA VENTA LA 1ª REVISTA INTERNACIONAL PARA USUARIOS DE INTERNET



GRATIS
CD-ROM
QUE CONTIENE TODOS LOS PROGRAMAS NECESARIOS PARA CONECTARSE A INTERNET

ESPECIAL: INTERNET "RED DE REDES"

- **PROVEEDORES DE INTERNET: ¿Qué es un servidor FTP?**
- **EL CYBER-PROFESOR: Descripción del software necesario para acceder a Internet**
- **HERRAMIENTAS DE INTERNET: La conexión multimedia**
- **EL TURISTA VIRTUAL: Cómo hacer turismo sin salir de casa**
- **USENET: La vía para hacer llegar tu opinión a todo el mundo**
- **GLOSARIO DE TÉRMINOS**



100 nanofaradios, un jack de audio mono hembra, un conector macho de 25 pines para el puerto paralelo, una placa para montar el circuito y algo de maña con el soldador, puede conseguir obtener sonido de 8 bits con notable calidad. Sin embargo, dado que no hay memoria propia, ni control DMA, la reproducción mediante este dispositivo implica el uso intenso de la CPU, produciendo una carga importante en el sistema.

Como el sistema VoxWare que se encuentra con las fuentes del núcleo no incluyen ningún controlador para este hardware, hay un programa que se obtiene por separado, denominado pcsndrv, que permite reproducir sonido con este tipo de hardware, por muy poco coste.

Si usted no quiere tener nada que ver con este montaje, el cual siempre lleva un riesgo en caso de operar inadecuadamente, puede reproducir sonido a través del altavoz interno del PC, con un coste nulo, mediante el mismo paquete. En este caso, la calidad de sonido es claramente inferior, aunque la carga del sistema similar a la del conversor en puerto paralelo.

Descomprima el paquete en el directorio que desee (normalmente /usr/src ó /tmp). En primer lugar deberá aplicar los parches correspondientes al núcleo (que debe ser alguno de la versión 1.2.0 a la 1.2.11) y recompilarlo. La razón por la que este paquete no tiene perspectivas de ser incluido de forma oficial en Linux es la apreciación de Linus Torvalds de que este driver puede bloquear (literalmente) el sistema si se escoge una frecuencia de muestreo por encima de la potencia de nuestra CPU, ya que interfiere con el timer del sistema. Supuesto que se ha descomprimido en /tmp, el parche se aplica con:

```
cd /usr/src
patch -p0 </tmp/pcsnd/krn/pcsnd128.diff
cd /tmp/pcsnd/app
cp pcsp.h /usr/include/linux
```

Una vez recompilado el núcleo y rearrancado el sistema, tendremos accesibles nuevos dispositivos: /dev/pcaudio, /dev/pcmixer y /dev/pcsp que equivalen a los /dev/audio, /dev/mixer y /dev/dsp descritos anteriormente. Si no tiene tarjeta de sonido, puede sustituir unos ficheros por otros, para que las aplicaciones de sonido se puedan emplear sin cambios:

```
cd /dev
mv audio audio.hw
ln -s pcaudio audio
mv mixer mixer.hw
ln -s pcmixer mixer
mv dsp dsp.hw
ln -s pcsp dsp
```

Estos comandos no son necesarios si usted ya dispone de tarjeta de sonido. Teniendo disponibles ambas familias de dispositivos, usted podrá elegir por donde desea escuchar el sonido.

Tras recompilar los ejecutables que acompañan a esta distribución (deberá tener make, gcc y los "includes" adecuados instalados), los ficheros del directorio /dev se crean con el programa pcpsinstall, y se le puede asociar un hardware determinado (el altavoz, o un puerto paralelo) al dispositivo de sonido mediante el programa pcsel, también incluido. Por ejemplo

```
pcsel -d DACm
pcsel
```

Con el anterior ejemplo, pasamos del sonido por el altavoz a seleccionar el DAC (mono) del puerto paralelo, el cual Linux no puede detectar por sí mismo en el arranque. Con el segundo comando, puede observar el hardware actualmente seleccionado. Para volver a tener activo el altavoz, teclee:

```
pcsel -d PCSP
pcsel
```

Tras esto, todos los lectores deben estar en disposición de aprovechar el sonido en Linux, tanto si tienen tarjeta de sonido como si no, para poder programar con VoxWare. Como curiosidad final, en este paquete se incluye un fichero que, añadido a uno existente en nuestro sistema, permite reconocer mayor número de ficheros. Teclee lo siguiente:

```
cd /tmp/pcsnd/app
cat etc.magic >>/etc/magic
```

Una vez hecho esto, podrá reconocer ficheros MID, VOC y WAV. Observe los resultados del comando (suponiendo que la particion DOS está montada, y Windows instalado):

```
file /dosc/windows/tada.wav
```

El hardware de nuestra tarjeta de sonido libera de carga a la CPU

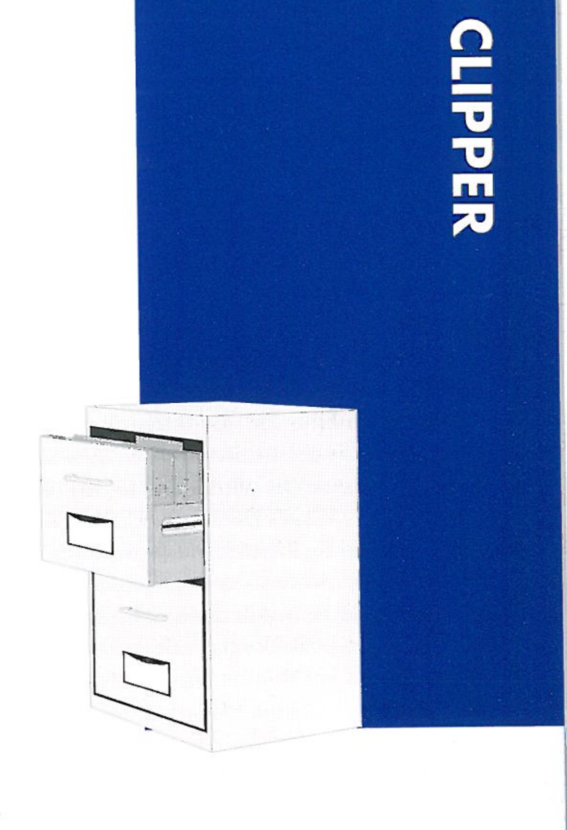
CONCLUSIONES

En una primera toma de contacto, podemos ver que el sonido en Linux está estandarizado, mediante un conjunto de llamadas que independizan el hardware de nuestro programa. Por otra parte, se nos proporciona un controlador para poder programar aplicaciones que manejen el sonido teniendo un hardware de coste ínfimo (DAC en puerto paralelo) o nulo (el altavoz interno). También hemos visto que toda la ayuda disponible, como transferencias DMA, bufferes amplios, hardware específico para generación de sonido, y otros, ayudan a descargar la CPU, factor imprescindible para un funcionamiento fluído de un sistema multitarea.

En una próxima entrega, examinaremos el sistema VoxWare desde el punto de vista de la programación. Hasta entonces.

CONTROLADORES SUSTITUIBLES DE BASES DE DATOS

Juan Manuel y Luis Martín



En anteriores artículos se ha explicado en qué consisten las bases de datos, así como las partes que las componen. También se analizó la forma de manejarlas dentro de un programa realizado con Clipper. En el presente artículo se explicará la forma de utilizar bases de datos con un formato diferente al estándar de Clipper dentro de un programa realizado en Clipper.

Desde que Ashton Tate lanzó al mercado de la microinformática su famoso gestor de bases de datos dBASE, el mundo de las bases de datos para ordenadores personales ha sufrido una expansión que ha llevado a distintos fabricantes a crear los suyos propios. Esta evolución del mercado ha provocado la aparición de múltiples formatos de almacenamiento de los datos contenidos en las bases de datos, trayendo consigo una falta de compatibilidad. De esta forma, las bases de datos creadas con un determinado programa, como FoxPro, dBASE, Clipper, Paradox, etc., no podian ser utilizadas por el resto.

De entre todos los formatos existentes, Clipper utiliza el denominado DBFNTX, es decir, ficheros con extensión .DBF para las bases de datos y ficheros con extensión .NTX para los índices. La diferencia entre este formato y el resto, no sólo radica en el nombre de los ficheros, sino que además, su estructura interna también es diferente.

Es a partir de la versión 5.1 de Clipper cuando surge la idea de posibilitar el uso de diferentes formatos de bases de datos en un mismo programa. Esto es posible mediante la utilización de los denominados RDDs (Replaceable Database Driver o Controlador Sustituible de Bases de datos).

Por tanto, un RDD puede definirse como una libreria suministrada por Clipper en la que se incluye la implementación de las funciones y mandatos orientadas al manejo de un formato concreto de bases de datos, así como de sus índices. De esta forma, todas las funciones y mandatos vistos en artículos anteriores se encuentran incluidos en la librería DBFNTX.LIB. Esta librería es enlazada de forma automática con los todos aquellos programas que utilizan bases de datos.

Como se ha indicado anteriormente, con la versión 5.1 del lenguaje aparece esta nueva forma de trabajo, aunque en dicha versión únicamente se incluía la librería DBFNTX.LIB. Es a partir de la versión 5.2 cuando, continuando con esta filosofía, se incluyen en el paquete una serie de librerías adicionales que permiten al programador crear aplicaciones en Clipper capaces de manejar bases de datos de distintos formatos.

INCLUSIÓN DE RDD´s EN UN PROGRAMA

Como se ha mencionado anteriormente, Clipper trabaja por defecto con la librería DBFNTX.LIB. Esta libreria es cargada de manera automática por el propio compilador y enlazador, sin necesidad de realizar ninguna de las operaciones que serán descritas a continuación.

Para poder utilizar cualquier otra librería deben realizarse una serie de operaciones. La primera de ellas consiste en indicar la carga en memoria de la librería que se va a utilizar. Esta operación se realiza mediante un mandato estándar de Clipper cuya sintaxis es:

REQUEST Librería

Con esta sentencia se le indica al compilador que ciertas llamadas a funciones pueden resultarle desconocidas, pero que el módulo que contiene dichas funciones será enlazado con el fichero ejecutable final.

Si se va a utilizar más de un controlador RDD, deberá aparecer una senten-

cia REQUEST para cada uno de ellos. Además, esta sentencia debe ir situada antes que cualquier otra sentencia ejecutable del programa. También debe tenerse en cuenta que la inclusión de esta sentencia en un programa no exime de la necesidad de incluir la librería del controlador en la sentencia LIBRARY del enlazador durante el proceso de enlace del programa.

EL ARCHIVO RDDSYS.PRG

En el sistema de Clipper se incluye un archivo especial encargado de la configuración interna de los controladores RDD. Se trata del archivo RDDSYS.PRG, que se encuentra almacenado en el directorio \CLIPPER5\SOURCE\SYS. Este archivo es incluido de forma automática en el proceso de compilación, y su contenido se muestra en el listado 1.

En dicho listado caben destacar dos operaciones realmente importantes. La primera de ellas consiste en la carga en memoria mediante la sentencia REQUEST de la librería que contiene la especificación del controlador DBFNTX, tal y como se ha explicado anteriormente:

REQUEST DBFNTX

La otra operación permite especificar que dicho controlador será el que se utilice por defecto en el programa. Esta operación se realiza mediante la función rddSetDefault(), encargada de indicar al sistema cual será el controlador a utilizar con todas aquellas base de datos para las que no se especifique ningún controlador .

rddSetDefault("DBFNTX")

Al igual que sucede con otros ficheros proporcionados por el sistema de Clipper, este fichero puede ser alterado, permitiendo así cambiar la configuración interna del sistema de controladores RDD. De esta forma, mediante la inclusión de sentencias REQUEST adicionales, es posible indicar la carga en memoria de más librerías. Igualmente, es posible alterar el controlador RDD que va a ser utilizado por defecto.

Si se realiza alguna variación en dicho fichero, será necesario realizar la inclusión del mismo en nuestro programa de forma explícita. Para ello, habrá que compilar el nuevo fichero RDDSYS.PRG para obtener el nuevo módulo objeto RDDSYS.OBJ. Posteriormente será necesario realizar el enlace del nuevo módulo objeto obtenido con el módulo objeto resultante de la compilación del programa. En resumen, los pasos a seguir si se altera el fichero RDDSYS.PRG serían los siguientes:

C:\>clipper rddsys
C:\>clipper programa
C:\>rtlink fi programa rddsys li <librerías de RDDs utilizados>

UTILIZACIÓN DE UN CONTROLADOR

Una vez explicada la forma en que pueden realizarse las operaciones de carga y especificación de los controladores RDD que van a ser utilizados en un determinado programa, es el momento de analizar la forma en que pueden ser utilizados.

Como se explicó en artículos anteriores, para el trabajo con varias bases de datos Clipper realiza una especie de división de la memoria del ordenador en las denominadas áreas de trabajo. En cada una de estas áreas, únicamente podrá abrirse una base de datos con todos sus ficheros asociados (ficheros de índice, de campos memo, etc.). De esta forma, cuando se realiza la apertura de una base de datos, Clipper asigna el controlador encargado de gestionar dicha base de datos al área de trabajo en la que se abre. Así, todas las operaciones que afecten a la base de datos de esa área serán gestionadas por las funciones implementadas en la librería que contiene la especificación de dicho controlador.

La ventaja que aporta este sistema es que todas las funciones de manipulación de bases de datos vistas en anteriores artículos están implementadas en todos los controladores. De esta forma, el programador no debe preocuparse del formato de la base de datos que esté utilizando, ya que las operaciones se realizan mediante las mismas funciones, siendo el controlador el encargado de implementar dichas operación en función del formato de la base de datos.

Sólo existe un momento en el que se puede especificar el controlador a utilizar en un área de trabajo determinada. Como el lector podrá suponer, se trata del momento de la apertura. Por ello, tanto el mandato USE como la función DBUSEAREA(), ambos encargados de la apertura de bases de datos, incorporan una cláusula en la que puede realizarse la especificación del controlador. Además, debe tenerse en cuenta que una base de datos también se abre en el momento de su creación por lo que también el mandato CREATE y la función DBCREATE() incorporan cláusulas o parámetros para la especificación del controlador. La sintaxis de estos mandatos y funciones puede consultarse en el cuadro 1.

Una vez abierta una base de datos y especificado su controlador, la manipulación de la misma se realizará mediante las mismas funciones vistas hasta ahora, salvo pequeñas variaciones propias de cada formato, que se serán analizadas un poco más adelante. Por lo general, todas las funciones vistas hasta el momento en anteriores artículos funcionan de la misma forma, sin importar el formato de la base de datos sobre la que son utilizadas.

Una vez realizadas las operaciones necesarias, la base de datos debe de ser cerrada, utilizando para ello los mandatos y funciones estándar vistos con anterioridad (el mandato USE sin argumentos o las funciones DBCLOSEALL() o DBCLOSEAREA()). La ejecución de estas sentencias supone la liberación total del área de trabajo, es decir, tanto el cierre de la base de

LISTADO 1

```
/***
*
* RddSys.prg
*
* This program is run each time your
* application is started to setup
* the proper default driver.  You should
* modify a copy of this file if you wish
* to use a default RDD other than DBFNTX.
*
* Copyright (c) 1993,
* Computer Associates International, Inc.
* All rights reserved.
*
*/

// This line must not change
ANNOUNCE RDDSYS

INIT PROCEDURE RddInit

  // Causes DBFNTX RDD to be linked in
  REQUEST DBFNTX

  // Set up DBFNTX as default driver
  rddSetDefault( "DBFNTX" )

RETURN

// eof: rddsys.prg
```



datos y sus ficheros asociados como la desasignación del controlador. De esta forma, es posible volver a utilizar dicho área de trabajo para abrir otra base de datos con un formato distinto.

A continuación se explicarán las principales características de los controladores sumunistrados con Clipper, con el fin de que el programador pueda tener en cuenta las peculiaridades de cada uno de ellos, así como las pequeñas variaciones que pueden existir en su utilización. La tabla 1 muestra una lista de los controladores proporcionados por Clipper.

EL CONTROLADOR DBFNTX

Como se ha venido mencionando a lo largo del artículo, éste es el controlador que utiliza Clipper por defecto. Algunas de sus principales características, que lo diferencian del resto de controladores, son las siguientes:

- Los archivos de bases de datos son compatibles con el formato xBase.
- Permite la creación de índices condicionales, mediante la utilización de las cláusulas de ámbito, FOR y WHILE.
- Permite la creación de un índice aunque exista otro abierto y activo.
- Permite el control del proceso de indexación mediante las cláusulas EVAL y EVERY.
- Permite la creación de índices descendentes.

EL CONTROLADOR DBFNDX

Este controlador permite la utilización de base de datos creadas mediante los programas dBASE III y dBASE III PLUS. Este formato aporta ficheros de bases de datos con la extensión .DBF y ficheros de índice con la extensión .NDX. El funcionamiento de todos los archivos asociados, incluido el propio archivo de base de datos, es idéntica al formato BDFNTX. Tan sólo existen dos particularidades relacionadas con los ficheros de índice, que son las siguientes:

- Cuando se crea un índice con este controlador, no está permitida la inclusión de campos lógicos dentro de la expresión clave. Para poder utilizar este tipo de campos es necesario convertirlos previamente en cadenas de caracteres.
- La expresión clave sólo puede estar formada por funciones compatibles con dBASE III PLUS, ya que de otra forma, el archivo de índice no será reconocido por dBASE III PLUS, aunque sí por el propio Clipper.

Además de estas consideraciones, existe una limitación en la utilización compartida de este controlador en entornos de red. De esta forma, no es posible realizar accesos concurrentes a una misma base de datos desde programas realizados en Clipper que utilizan este controlador y el propio dBASE III PLUS.

EL CONTROLADOR DBPX

Este controlador permite la utilización de bases de datos creadas con el programa Paradox. Su principal característica es que los archivos de bases de datos de este formato no siguen el estándar xBase, es decir, el archivo de base de datos no tiene la extensión .DBF, sino la extensión .DB. Esto implica que la forma de almacenamiento de los datos también es diferente.

La otra gran diferencia radica en los ficheros de índice asociados a la base de datos. Para ellos existen dos formatos diferentes con distintos métodos de creación.

El primero de ellos consiste en la creación del índice en el mismo momento en el que se crea la base de datos. Para realizar esta operación bastará con añadir un asterisco al final del nombre del campo que se desea que forme parte de la clave en el momento de la especificación de la estructura de la base de datos. Por ejemplo, para crear una base de datos con dos campos, un nombre y una dirección, en la que se desea que los registros estén ordenados por el nombre, basta realizar las siguientes operaciones:

```
aBase := { { "Nombre*", "C", 20, 0 },;
 { "Direcc", "C", 30, 0 } }
DBCEATE( "clentes", aBase, "DBPX" }
```

De esta forma, se crean dos ficheros. Uno de ellos, con el nombre CLIENTES.DB, será el fichero que contendrá los datos propiamente dichos. El otro, con el nombre CLIENTES.PX, será un fichero de índice con la ordenación de la base de datos por el campo "Nombre". Este índice siempre será considerado el primero en la lista de índices abiertos, y se abre de forma automática en el momento de abrir la base de datos, sin necesidad de incluirlo en la cláusula INDEX del mandato USE.

CUADRO 1

```
USE [xcBaseDatos] [INDEX xcListaIndices] [ALIAS xcAlias]
  [NEW] [READONLY] [VIA xcControladorRDD]

DBUSEAREA( [lAreaNueva], [cControladorRDD], cBaseDatos, [xcAlias], [lCompartido], [lSoloLectura] )

DBCREATE( cBaseDatos, aEstructura, [cControladorRDD] )

CREATE xcBaseDatos FROM xcBaseDatosEstructural [NEW]
  [ALIAS xcAlias] [VIA cControladorRDD]
```

Sintaxis de mandatos y funciones de apertura y creación de bases de datos.

La otra posibilidad consiste en la creación de los índices posteriormente a la creación de la base de datos. Para ello, deben utilizarse los mandatos y funciones estándar de creación de índices de Clipper. El resultado de esta operación son dos ficheros de índice con el nombre indicado en la sentencia de creación y con las extensiones .X02 y .Y02. A medida que se vayan creando más índices, el número que contiene las extensiones irá aumentando (.X03 y .Y03, .X04 y .Y04, etc.). El aspecto más importante que debe tenerse en cuenta cuando se manejen bases de datos con este formato, es que cada uno de los índices estará compuesto por dos ficheros en lugar de uno, como era lo habitual en los otros formatos.

EL CONTROLADOR DBFCDX

Este controlador permite la utilización de bases de datos creadas con el programa FoxPro 2. El formato del fichero de bases de datos sigue el estándar xBase, por lo que el fichero tendrá la extensión .DBF, y su estructura y funcionamiento es idéntico al de otros formatos.

Sin embargo, este formato presenta dos importantes variaciones con respecto a los anteriores. Por un lado existe la posibilidad de utilizar los denominados archivos de índice compuestos. Básicamente, estos archivos son ficheros de índice que en lugar de contener un único índice pueden contener varios. En estos casos, el archivo de índice recibe el nombre de ORDER BAG, mientras que a cada índice en concreto se le denomina TAG. También pueden

TABLA 1	
Librería de Controlador RDD	Programa que lo utiliza
DBFNTX.LIB	Clipper
DBFCDX.LIB	FoxPro 2
DBFMDX.LIB	dBASE IV
DBFNDX.LIB	dBASE III y dBASE III PLUS
DBPX.LIB	Paradox 3.5

Controladores suministrados por Clipper en la versión 5.2.

utilizarse ficheros de índice simples, como los vistos hasta ahora.

La existencia de este tipo de ficheros hace que la manipulación de los índices varíe con respecto a los formatos anteriores. La primera de las variaciones se encuentra en el nombre del propio fichero de índice. Si el fichero contiene un único índice (fichero de índice simple) la extensión será .IDX. Sin embargo, si el fichero contiene varios índices (fichero de índices compuesto) la extensión será .CDX.

Dentro de un fichero de índices compuesto pueden almacenarse hasta 99 índices simples, aunque en la práctica se recomienda no pasar de 50. Por tanto, este controlador va a permitir sobrepasar el límite de índices abiertos para una base de datos, ya que, aunque sólo puedan tenerse 15 ficheros de índice abiertos a la vez, cada uno de ellos puede contener hasta 99 índices simples.

Para la creación de un índice compuesto basta con añadir una nueva cláusula al mandato INDEX ON. Esta cláusula permite indicar el nombre del índice de que se trata.

INDEX ON expClave [TAG cNomÍndice] TO xcNomFichÍndice

De esta forma, cada uno de los índices contenido en un fichero de índices compuesto será identificado por un nombre, que será el que se indique en la cláusula TAG.

Además, existen otras variaciones relativas a las funciones de apertura y adición de índices a la lista de índices abiertos de la base de datos. Para ello, Clipper incorpora una nueva función que viene a sustituir a DBSETINDEX().

ORDLISTADD(cFichÍndice, [cNomÍndice])

Como puede deducirse, esta función permite añadir un índice simple a la lista de índices abiertos en el área de trabajo actual. Los parámetros que la acompañan especifican el nombre del fichero y del índice, respectivamente.

También se producen algunos cambios en la operativa de activación de un índice concreto. Para ello, puede utilizarse el mandato SET ORDER TO con nuevas cláusulas, o utilizar una función nueva aportada por Clipper. La sintaxis para ambos casos es la siguiente:

SET ORDER TO [nÍndice | [TAG cNomÍndice] [IN xcFichÍndice]]

ORDSETFOCUS([cNomÍndice | nÍndice], [xcFichÍndice])

Por último, la operación de eliminación de un índice del fichero de índices compuesto también se vera alterada. Los cambios en la operativa son similares a los del caso anterior.

DELETE TAG cNomÍndice [IN xcFichÍndice]

ORDDESTROY(cNomÍndice[, cFichÍndice])

Otra de los cambios importantes que trae consigo la utilización de este formato es el tratamiento de los campos memo. El fichero asociado que contiene la información de los campos memo tendrá la extensión .FPT. Además, esta organizado en bloques de 64 bytes en lugar de los 512 bytes del formato DBFNTX. Esto hace posible que se produzca un aprovechamiento más racional del espacio ocupado por dicho archivo, disminuyendo su tamaño en el disco. Las estimaciones de ahorro de tamaño rondan el 30% de reducción con ficheros .FPT, en relación a ficheros .DBT.

EL CONTROLADOR DBFMDX

Este último controlador permite la utilización de bases de datos con el formato utilizado por dBASE IV. Como la mayoría de los formatos, los ficheros de bases de datos siguen el estándar xBase, por lo que tendrán la extensión .DBF. Sin embargo, los ficheros de índice tendrán la extensión .MDX.

Otra variación importante que presentan los ficheros de índice de este formato es que todos pueden ser compuestos, igual que ocurría con los ficheros .CDX del formato anterior. El funcionamiento idéntico, por lo que deben usarse las mismas funciones y cláusulas que han sido explicadas para los ficheros .CDX del formato anterior. A diferencia de lo que ocurría con el controlador DBFNDX para bases de datos de dBASE III y dBASE III PLUS, este controlador proporciona una compatibilidad total con las bases de datos de dBASE IV. Esta compatibilidad llega hasta los accesos concurrentes a una base de datos en entorno de red entre aplicaciones realizadas Clipper que utilizan este controlador y el propio dBASE IV.

LAS NUEVAS FUNCIONES

Como se ha podido apreciar, la inclusión de los RDD en las últimas versiones de Clipper ha traído consigo la aparición de una serie de mandatos y funciones encaminadas a facilitar su utilización. Algunas de ellas ya han sido introducidas al presentar los controladores DBFCDX y DBFMDX. Sin embargo, existen muchas mas que pueden ser consultadas en la ayuda interactiva de Clipper, proporcionada por las Guías Norton. En general, la mayoría de estas funciones pueden identificarse fácilmente, ya que suelen comenzar con las letras ORD.

Además, en algunos de los mandatos y funciones ya existentes se han introducido algunas variaciones, incluyendo nuevas cláusulas para el manejo de los controladores. Así, por ejemplo, los mandatos APPEND FROM y COPY TO ahora disponen de la cláusula VIA, que permite especificar el controlador a utilizar.

CONCLUSIÓN

A lo largo de este artículo se ha proporcionado una primera aproximación a los controladores sustituibles de bases de datos. Como ha podido apreciarse, se trata de una herramienta realmente práctica. Con la inclusión de esta filosofía en Clipper se obtiene un nivel de abstracción que permite al programador la manipulación de distintos formatos de bases de datos sin necesidad de alterar sus aplicaciones.

Finalmente, es recomendable que el lector realice sus propias pruebas con los diversos controladores proporcionados por Clipper. Es de esperar que en futuras versiones se aumente el número de éstos, lo que permitirá utilizar la casi totalidad de formatos existentes.



JUEGOS CONVERSACIONALES

Ignacio Cea

Seguro que recuerda aquellos tiempos no tan lejanos en los que los simples juegos conversacionales copaban no sólo el mercado sino también nuestra imaginación. Juegos en los que podíamos hablar con el ordenador para intentar rescatar una reina o encontrar un diamante perdido en el tiempo. En esta serie de artículos le mostraremos como hacerlo en C++.

No tiene más que fijarse un poco para observar que la mayor parte de los clásicos juegos conversacionales comparten la misma forma de trabajo. De hecho, lo único que realmente los diferencia es el marco en el que se desarrollan y los objetos que intervienen dentro de él.

La serie de artículos que ahora comienza pretende aprovechar las características de generalización y abstracción inherentes a las nuevas tecnologías de orientación a objetos (y en concreto al C++) para conseguir un conjunto de clases que puedan ser usadas como base en la programación de cualquier juego conversacional.

EL PLAN

El total de artículos que vamos a dedicar a este tema, podemos dividirlo en cuatro grandes bloques: En el primero, de descripción, trataremos de identificar las entidades comunes a cualquier aplicación de este tipo. En el segundo, de desarrollo, pasaremos a la programación de cada una de ellas con el fin de incluirlas en una librería y en el tercero, y último, desarrollaremos un ejemplo completo basándonos en dicha librería.

En cada una de esas partes introduciremos conceptos o bien hasta ahora desconocidos o bien lévemente tratados. De esta forma, la primera fase ahondará en términos y símbolos propios de las metodologías de modelado de objetos en grandes proyectos. La segunda, por otro lado le introducirá a la persistencia de objetos (grabar objetos en disco) mientras que la última le descubrirá, en todo su esplendor, el reuso de código.

EL MÉTODO

Con el tiempo comprenderá que las aplicaciones desarrolladas en C++ y, en general, empleando técnicas de orientación a objetos necesitan un tiempo de análisis superior al que las clásicas funcionales necesitaban.

El emplear técnicas de orientación a objetos para el desarrollo de un programa no garantiza que el resultado vaya a ser mucho mejor de cara al futuro que el obtenido con técnicas más vetustas. Es necesario pensar, desde el principio en el reuso, en la distribución de responsabilidades y en esas otras tantas cosas que ya ha escuchado a lo largo de esta colección de artículos.

Toman, por tanto, especial interés las metodologías de desarrollo de aplicaciones bajo estas técnicas: Esas metodologías no son más que un conjunto de pasos y documentos a presentar en cada uno de ellos para, partiendo de las especificaciones iniciales de un proyecto llegar a una solución final que reúna, sino todas la mayor parte, de las ventajas de la orientación a objetos.

El exponer alguna de estas metodologías no es el objetivo de la serie de artículos que ahora se inicia; sin embargo si que consideramos oportuno esbozarle unas pinceladas de los pasos a seguir con el fin de que disponga de un hilo conductor en el caso de que su curiosidad le lleve a aplicarlas en sus propios sistemas.

CICLO DE VIDA ITERATIVO

Los proyecto clásicos transcurrían siempre de forma secuencial entre las fases de análisis, diseño e implementación y, si bien la vuelta hacia atrás era continuamente necesaria y real, no estaba contemplada como parte integrante del proceso.

La orientación a objetos, por el contrario, si lo hace y aprovecha esas continuas idas y venidas para crecer



Figura 1. Modelo de objetos.

de forma más cercana a las necesidades del proyecto.

Si los métodos clásicos eran secuenciales, los orientados a objetos son cíclicos.

EL ANÁLISIS

Inicialmente tiene lugar una fase de captación de requerimientos, en la que, como en los proyectos clásicos se intentan capturar las necesidades del sistema a construir.

Seguidamente, se entra en la fase de análisis donde la actividad principal consiste en extraer de la anterior descripción las entidades u objetos que participan en el problema, sus atributos, las acciones que han de emprender y las relaciones que los unen con otros.

El resultado de esta fase es algo, independiente del lenguaje final de programación.

EL DISEÑO

La fase de diseño trata de llevar a un ordenador el conjunto de objetos antes pensados; se tienen en cuenta cosas como la persistencia de objetos, el manejo de errores, los tipos de los atributos, la forma de implementar las relaciones, etc...

El resultado de esta fase son, nuevamente, objetos que son representados dentro de diagramas en todo análogos a los obtenidos en la primera etapa.

Es más, posiblemente se detecte la necesidad de retoques dentro del análisis. Con este método puede volverse siempre que se quiera hacia atrás.

LA IMPLEMENTACIÓN

La implementación con lenguajes orientados a objetos como C++ o SmallTalk no es más que la transcripción literal de lo que estará reflejado dentro de los diagramas surgidos durante las fases de análisis y diseño.

Si, por el contrario, decidiera emplear otro lenguaje más clásico como C, habría de autoestablecerse unas normas generales para transcribir las clases a estructuras de datos y funciones.

Dentro de esta fase es, también, normal encontrarse con imprevistos que hagan necesaria la vuelta atrás, incluso hasta la fase de análisis.

IDAS Y VENIDAS

El hecho de que la información de un problema esté muy encapsulada dentro de las clases hace que las continuas idas y venidas del método entre las distintas fases no tengan una repercusión tan grande como en un principio se podría pensar.

EQUIVOCARSE ADREDE

Ni que decir tiene que esto es algo que no se ve hasta que haber tratado con un ejemplo concreto. Por eso aprovecharemos este ejemplo para mostrarle esos conceptos. En una palabra: nos vamos a equivocar aposta.

NUESTRO EJEMPLO

Nuestro ejemplo va a seguir todas las etapas del desarrollo antes mencionadas pero sin detenerse en el detalle de cada una de ellas pues no es el objetivo de la serie.

Podríamos considerar que la descripción de requerimientos es el párrafo de la introducción. Por tanto hemos de comenzar por identificar las clases que participan dentro del problema.

COMO EMPEZAR

Para tratar de identificar que entidades intervienen en el problema a resolver hemos de fijarnos sólo en la realidad del mismo, olvidando problemas inherentes a la implementación de la solución tales como: formas de almacenar la información del juego, formas de comunicación del usuario, sonidos, gráficos, etc...

Esas cosas, sin lugar a dudas importantes, han de ser consideradas posteriormente. Lo único que debe ahora preocuparnos es quien interviene (clases), que atributos les caracterizan y de que forma se relacionan con las demás clases.

Cada una de estas ideas puede ser mostrada de muy diversas maneras: desde un simple texto descriptivo hasta complejos grafos en los que, por ejemplo, las fotos de las entidades

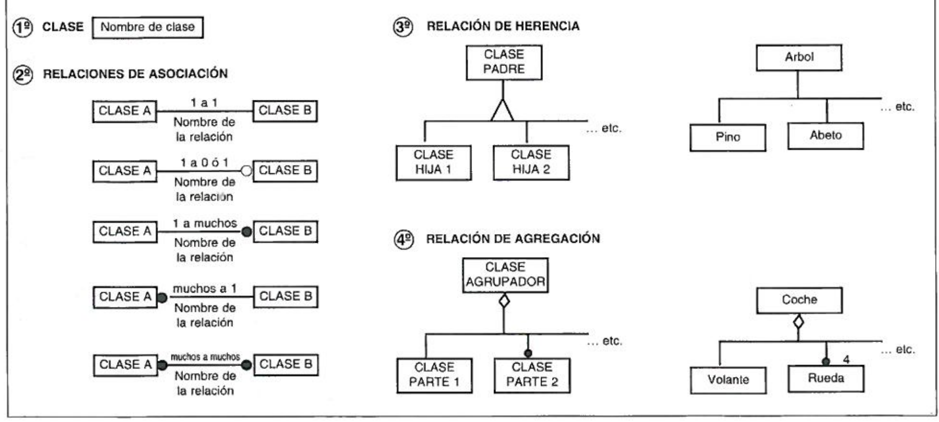


Figura 2. Notación ONT.

reales aparezcan unidas a otras mediante flechas (figura 1). Nosotros vamos a emplear la notación establecida por James Rumbaugh en su método OMT. Los aspectos más importantes de esta notación puede verlos en la figura 2.

Es necesario destacar que la metodología OMT es un método completo para el desarrollo de aplicaciones orientadas a objeto, por supuesto mucho más extensa que las simples ideas que aquí se exponen. No es, sin embargo, objeto de la serie la exposición de la mismo y la usaremos sólo para excitar su curiosidad y para apoyar la deducción de clases que se realizará al final.

LAS ENTIDADES

Encontrar las clases que intervienen en un problema es algo que se simplifica conforme aumenta la experiencia. Sin embargo, un buen camino para empezar puede ser, por ejemplo, describirse a uno mismo y de forma textual las características más importantes del problema tratado. Veamos como sería en nuestro caso.

La descripción puede ser la simple descripción de las características de un juego cualquiera; sin embrago, no hemos de olvidar que nuestro objetivo último son crear unas clases comunes a todos los juegos y precisamente sobre ese tema hemos de centrar nuestra atención.

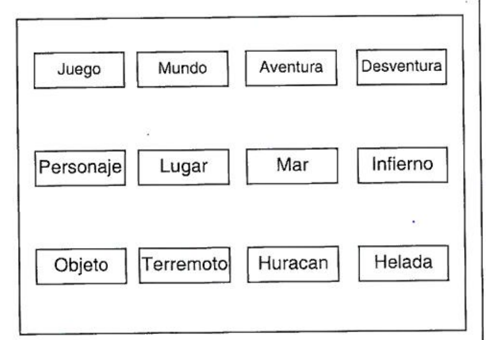


Figura 3. Las clases iniciales.

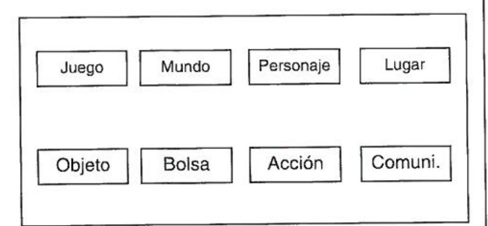


Figura 4. Las clases finales.

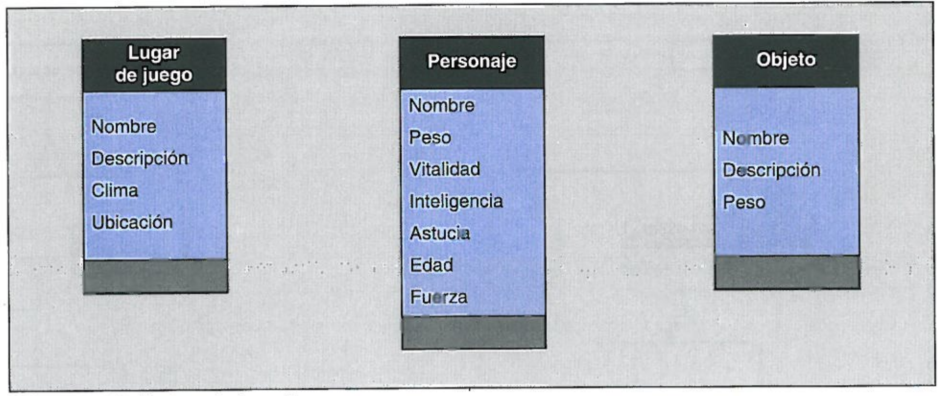


Figura 5. Atributos de las clases.

EL PROBLEMA

"Cualquier juego conversacional se desarrolla siempre sobre un mundo más o menos imaginario. Por ese mundo es por donde tienen lugar las aventuras y desventuras de los personajes del juego. Es normal que dentro de ese mundo encontremos diversos lugares; algunos bellos como el mar y otros desagradables como el infierno. En nuestro vagar podremos hablar, preguntar, guerrear y, en general, relacionarnos con todo aquellos que veamos: objetos, personajes, etc...

El mundo estemos o no moviéndonos, gira y gira y puede deparanos sorpresas como terremotos, huracanes, heladas, etc.."

No cabe duda de que el anterior párrafo es una descripción bastante genérica de lo que es un juego conversacional. Eso es, precisamente, lo que queremos ya que nos permitirá extraer de él clases completamente reusables entre distintos juegos.

LAS CLASES

Las clases suelen ser los nombres comunes que aparecen dentro de la anterior descripción. La mejor manera de reconocerlos es, sin duda, la experiencia, pero a falta de pan utilice la anterior regla.

De esta forma, podríamos deducir que clases son aquellas que aparecen en el figura 3.

Ya que es nuestro objetivo conseguir clases generales podemos olvidarnos de aquellas que parezcan excesivamente particulares; tales como: terremoto, huracán, mar e infierno. De tal forma que una buena lista inicial podría ser la de la figura 4.

LOS ATRIBUTOS

La elección de los atributos debe seguir una pauta similar a la de las clases. Es decir, debemos fijarnos sólo en aquellos genéricos a todos los juegos. Los que nosotros consideremos por cada clase se muestran en la figura 5.

En esta primera fase sólo necesitamos obtener una idea genérica de la información necesaria para describir todos los objetos considerados. No hemos de fijarnos, por tanto, en temas como el tipo de los atributos o su visibilidad. Eso es algo que aparecerá más adelante.

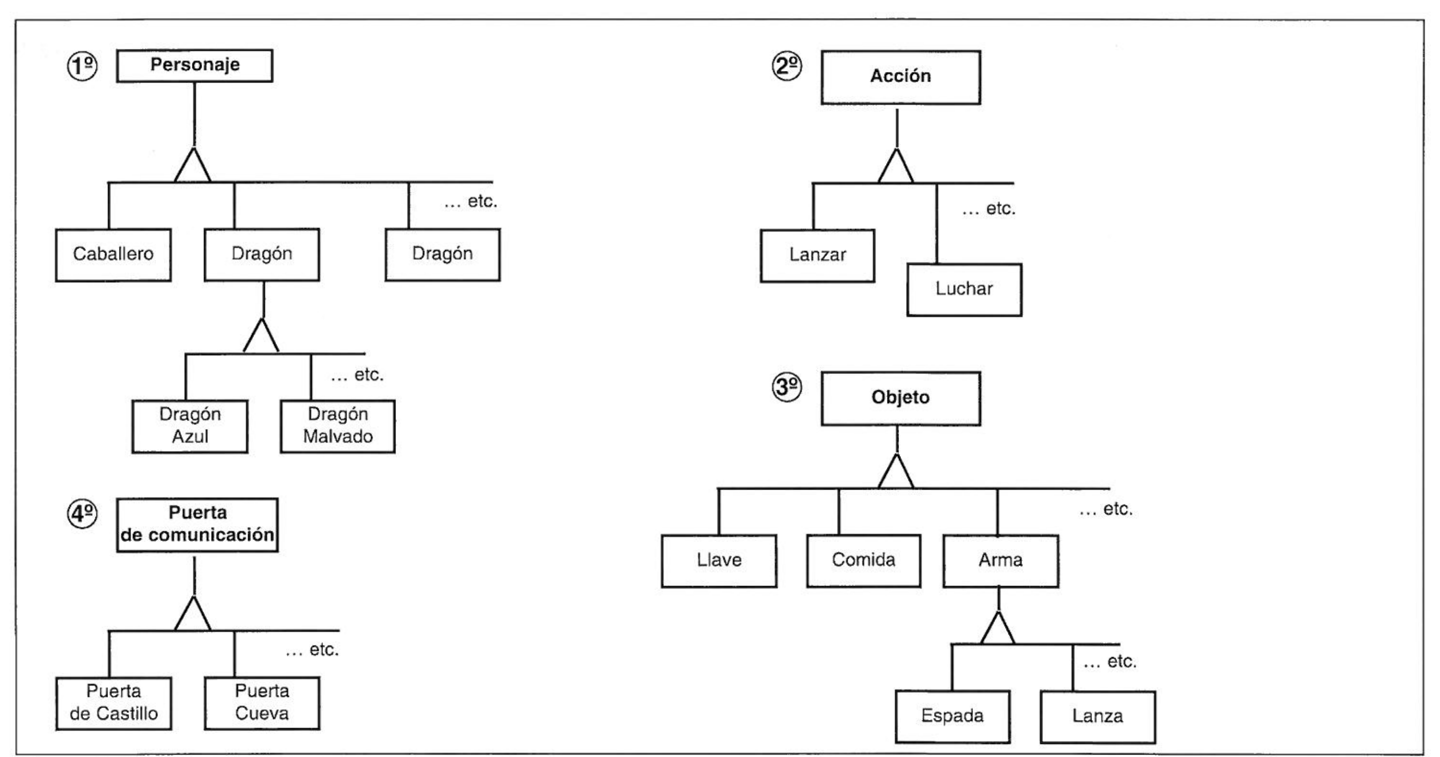


Figura 6. Detallando los objetos de un juego concreto.

ADAPTÁNDOLO A UN JUEGO CONCRETO

Es muy probable que su juego necesite más información que la que aquí vamos a considerar. Pero para ello tiene el recurso más empleado es el de la herencia. Es decir, le bastaría crear una clase hija de la que desea ampliar añadiéndole sólo aquellas características que la otra no incorpora. Sin duda que la cantidad de código que finalmente tendrá que escribir será mucho menor. Vea un ejemplo de ello en la figura 6.

En esa figura se muestran las clases que habría que añadir de cara a desarrollar un juego conversacional ambientado en la época medieval: Los caballeros, los dragones y las princesas seguirían siendo tipos de personajes pero con unas características (atributos) y comportamiento (métodos) más complejos y detallados que los descritos en la clase padre.

LAS RELACIONES

Un aspecto muy importante, y no mostrado hasta ahora, son las relaciones que unen a todas las clases consideradas. Esas relaciones nos permitirán saber de que forma van a colaborar entre si para solucionar cualquier acción del juego. El modelo de objetos completo se muestra en la figura 7.

UN CONSEJO

Antes de seguir adelante, puede ser buena práctica el plantearse cualquier otra pequeña aplicación y graficar un modelo semejante al de la figura 7 con el único fin de adquirir algo más de destreza en al análisis de problemas. Pruebe, por ejemplo, con el sistema de control para un lavado automático de coches.

Al final del artículo se recomiendan algunos libros que versan sobre las metodologías más importantes de análisis y diseño. En ellos podrá ampliar enórmente los conocimientos que aquí, tan sólo, se enuncian.

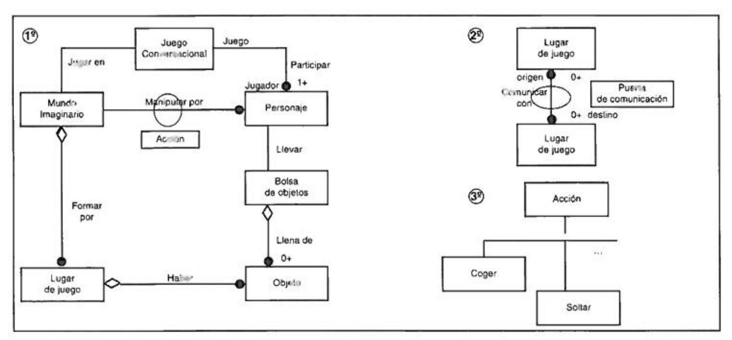


Figura 7. Modelo de objetos completo.

NO ES TAN FÁCIL

Las clases que hemos identificado tras la descripción del problema son las denominadas, dentro de la mayor parte de las metodologías, como semánticas. La librería final que generemos estará formada por muchas otras necesarias para llevar a cabo su implementación en un ordenador. Esas clases aparecerán en artículos posteriores, a la par que estas, las semánticas, van adquiriendo más nivel de detalle.

LA PRÓXIMA ENTREGA

En la próxima entrega vamos a ocuparnos de los métodos de cada una de las clases y escribiremos las primeras clases de aplicación requeridas.

REFERENCIAS

"Object Oriented Modeling and Design". James Rumbaugh. Editorial Prentice Hall (1991).

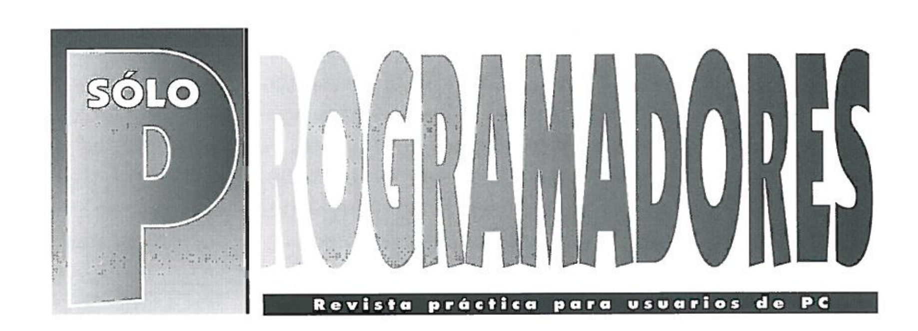
"Object Oriented Design with Applications". Grady Booch. Editorial The Benjamin/Cummings Publishing Company (1991).

CORREO LECTORES

Para cualquier duda referente a los temas/artículos tratatos en la revista, escriba una carta a:

Sólo Programadores
Referencia: *Correo Lectores*
TOWER COMMUNICATIONS
C/ Marqués de Portugalete, 10 Bajo
Madrid 28027

COMO SUSCRIBIRSE A



uscríbase enviando este cupón por correo o fax (91) 320.60.72, o llamando al teléfono (91) 741.26.62 Horario 9 a 14 y 15:30 a 18:30 h.

)eseo suscribirme a la revista SÓLO PROGRAMADORES acogiéndome a la siguiente modalidad:

☐ Suscripción: 1 año + regalo (filtro monitor) **por sólo** 10.995 ptas. ☐ Estudiantes carreras técnicas 40% Dto.: 8.250 ptas.

\lombre y apellidos...Domicilio...

'oblación...C.P ....Provincia........................Telf..............................Profesión...................

'ORMA DE PAGO:

☐ Con cargo a mi tarjeta VISA nº ....

Fecha de caducidad de la tarjeta................Nombre del titular, si es distinto...

☐ Domiciliación bancaria.

Señor Director del banco ....

Población ....

CODIGO CUENTA CLIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

Ruego a vd. que se sirva cargar en mi ☐ cuenta corriente ☐ libreta de ahorro

el recibo que le será presentado por TOWER COMMUNICATIONS, S.R.L. como pago de mi suscripción a la revista SÓLO PROGRAMADORES.

☐ Contra-reembolso del importe más gastos de envío.

☐ Cheque a nombre de TOWER COMMUNICATIONS S.R.L., que adjunto.

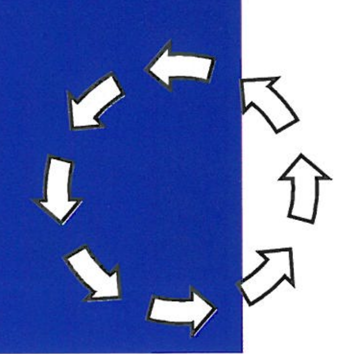
☐ Giro Postal (adjunto fotocopia del resguardo).

Firma:

Rellena este cupón y envíalo a:
TOWER COMMUNICATIONS SRL
C/ Marques de Portugalete 10, Bajo
28027 Madrid.

INCLUSIÓN DE GRÁFICOS Y USO DE MACROS

Enrique Castañón



En la entrega anterior vimos un ejemplo de ayuda interactiva. Se puso especial énfasis en el contenido de la ayuda, que debía contener y como exponerlo. Sin embargo no debemos descuidar la presentación. La inclusión de hypergráficos en la ayuda de una aplicación, además de resultar más agradable al usuario, le da una apariencia profesional. Se pueden crear ayudas visuales, en las que se muestren los cuadros de dialogo de la aplicación, y haciendo clic sobre las opciones aparezca una explicación de su uso. O incluso podremos crear hipermedios sobre cualquier tema en muy poco tiempo y con muy buenos resultados.

Las macros son también un elemento muy importante en el sistema de ayuda de Windows, son las que nos permiten personalizar los archivos de ayuda, y podemos, a la vez, definir nuestras propias macros. De forma que hay muy pocas cosas que no se pueden hacer, las pocas que hay, las resolveremos en una próxima entrega con la creación de Librerías de Enlace Dinámico (DLL), más bien estas limitaciones vendrán impuestas por nuestra imaginación.

El sistema de ayuda de Windows, nos permite crar hipermedios con una apariencia que no tendrá nada que envidiar a los creados con otras aplicaciones. Con muy poco esfuerzo y en muy poco tiempo podemos conseguir resultados sorprendentes.

TIPOS DE GRÁFICOS SOPORTADOS POR LA AYUDA

El sistema de ayuda de Windows soporta los siguientes formatos gráficos: bitmap estándar de Windows (BMP), bitmap independiente del dispositivo (DIB), meta-archivos de Windows (WMF), hypergráficos (SHG) y de múltiple resolución (MRB).

Hay una limitación importante a tener en cuenta, no se pueden presentar gráficos de más de 16 colores, y éstos deben tener la paleta de Windows, un gráfico con una paleta de 16 tonos de gris no se visualizará correctamente. Para poder presentar gráficos de 256 colores habría que crear una ventana filial (Embedded Windows) gestionada por una DLL. En una próxima entrega trataremos la creación de DLLs y ventanas filiales (Embedded Windows).

La utilidad MRBC.EXE combina en un solo archivo, gráficos de distintas resoluciones (CGA, EGA y VGA), y es el sistema de ayuda el que se encarga de mostrar la que se adapte a la resolución del monitor.

INCLUIR GRÁFICOS DIRECTAMENTE

Podemos insertar el gráfico en el archivo fuente (RTF) directamente, a través de una opción de menú, por ejemplo, en MS-Word 6, podemos utilizar la opción Imagen del menú Insertar, o copiarlo al portapapeles desde nuestra aplicación gráfica, y pegarlo en el texto con la opción Pegar del menú Edición, como se puede observar en la figura 1. Una limitación, es que no podemos insertar gráficos de más de 64K. Además si utilizamos un mismo bitmap en diferentes temas, habrá una copia de éste por cada tema, lo que incrementará el tamaño del archivo de ayuda. En ningún caso podremos incluir archivos de hipergráficos (SHG) empleando este método, tampoco los de múltiple resolución (MRB). No todos lo procesadores de texto crean un archivo (RTF) válido si insertamos directamente los gráficos, si trabaja

con MS-Word 6, no tendrá ningún problema. Una ventaja de usar este método para insertar gráficos, es que se trabaja viendo como va a quedar el archivo de ayuda definitivo.

INCLUIR GRÁFICOS POR REFERENCIA

También podemos hacer referencia al nombre del archivo gráfico en el texto del tema con la siguiente declaración: "{bmx nombre-bitmap}" donde "x" puede ser "l" para justificar a la izquierda, "r" para justificar a la derecha y "c" para centrar, nombre-bitmap puede ser cualquier archivo gráfico soportado por la ayuda. De esta manera podemos incluir gráficos de más de 64K, además, aunque se haga referencia a un mismo bitmap en diferentes temas, de éste sólo habrá una copia en el archivo de ayuda, con lo que reducimos el tamaño del archivo. Usar este método tiene el inconveniente de que es necesario compilar para ver el resultado, pero por contra no tendrá que editar el texto si desea cambiar o retocar el gráfico. Alternativamente podemos utilizar: "{bmxwd nombre-bitmap}", en este caso no podemos incluir gráficos de más de 64K, y habrá una copia del bitmap en cada tema que se incluya el gráfico. La figura 2 muestra el archivo RTF.

INCLUSIÓN DE HYPERGRÁFICOS.

Una opción interesante es que un gráfico puede representar un enlace con otro tema. El método es el mismo que emplearíamos para crear un hiperenlace con el texto, como ya se explico en la entrega anterior; subrayado doble del gráfico, e inmediatamente después, y con el formato de texto oculto, el identificador del tema al cual se quiere saltar. Más interesante aún es la posibilidad de segmentar el gráfico de forma que cada parte o segmento represente un enlace con un tema distinto. Para este tipo de hipergráficos necesitaremos la utilidad SHED.EXE, que crea archivos de hipergráficos segmentados (SHG). En el ejemplo que acompaña a este artículo podemos ver como hacer esto.

La utilidad SHED.EXE (Hotspot Editor) admite gráficos en formato

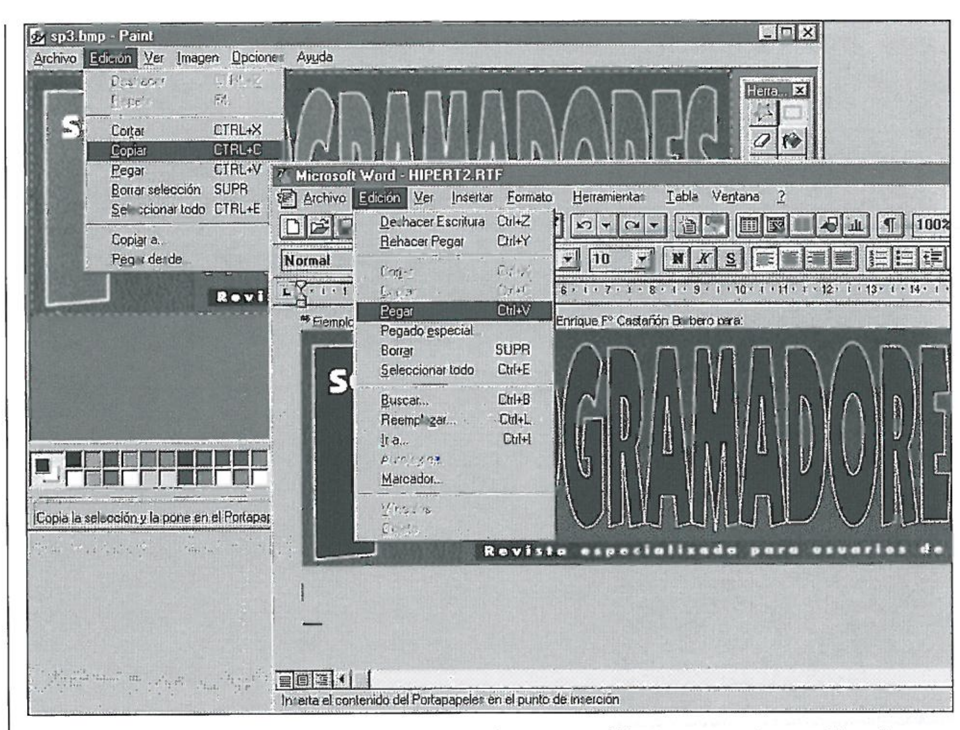


Figura 1. Mediante copiar y pergar podemos insertar gráficos en nuestro archivo de ayuda..

Aparentemente sencilla, la utilidad SHED.EXE, puede ser una potente herramienta

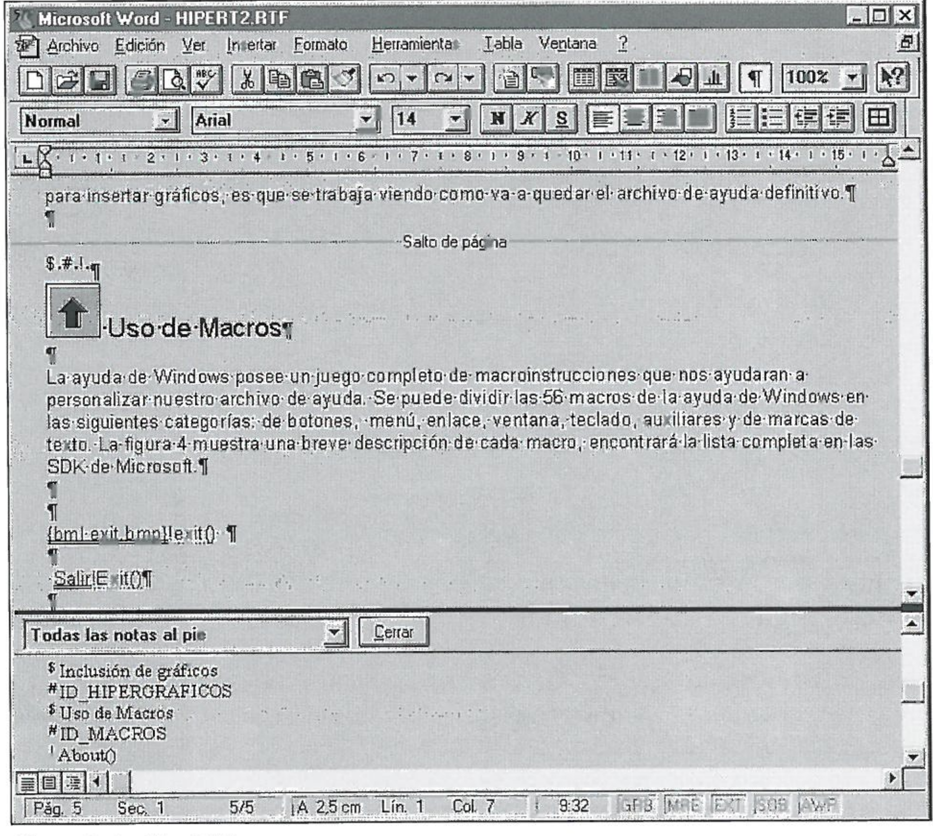


Figura 2. Archivo RTF.

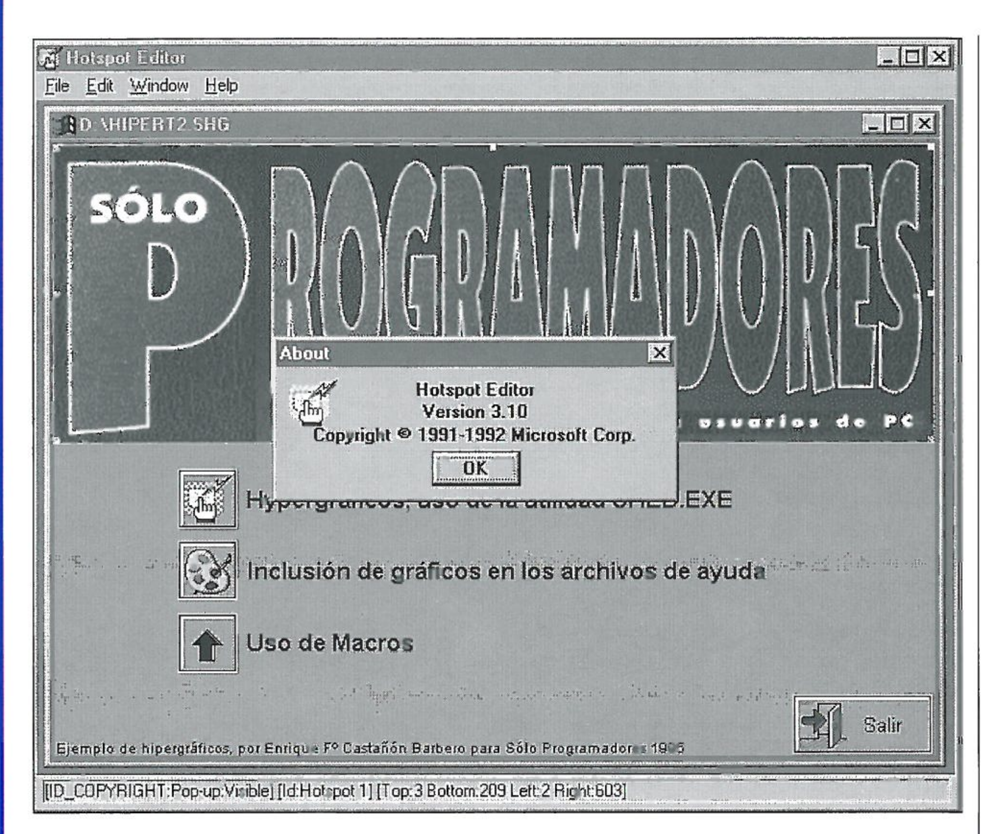
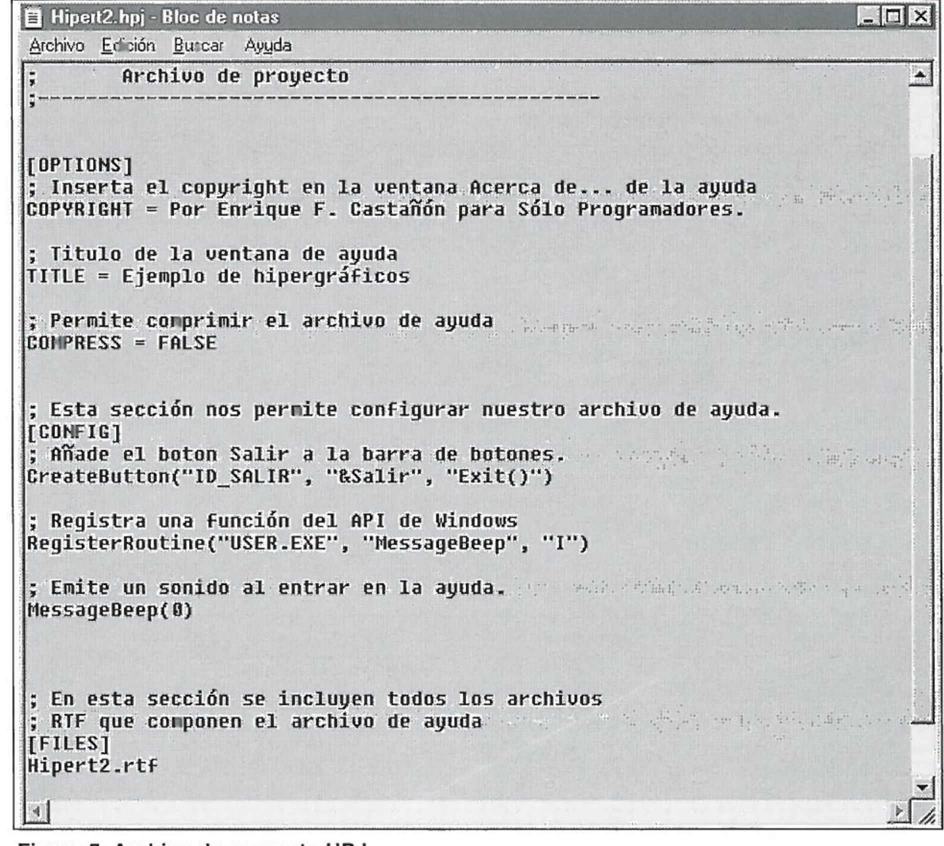


Figura 3. La utilidad SHED.EXE nos permite crear hipergráficos.

La única sección obligatoria del archivo de proyectos es [FILES]


Hipert2.hpj - Bloc de notas

Archivo Edición Buscar Ayuda

```
;       Archivo de proyecto
;----------------------------------------------


[OPTIONS]
; Inserta el copyright en la ventana Acerca de... de la ayuda
COPYRIGHT = Por Enrique F. Castañón para Sólo Programadores.

; Titulo de la ventana de ayuda
TITLE = Ejemplo de hipergráficos

; Permite comprimir el archivo de ayuda
COMPRESS = FALSE


; Esta sección nos permite configurar nuestro archivo de ayuda.
[CONFIG]
; Añade el boton Salir a la barra de botones.
CreateButton("ID_SALIR", "&Salir", "Exit()")

; Registra una función del API de Windows
RegisterRoutine("USER.EXE", "MessageBeep", "I")

; Emite un sonido al entrar en la ayuda.
MessageBeep(0)



; En esta sección se incluyen todos los archivos
; RTF que componen el archivo de ayuda
[FILES]
Hipert2.rtf
```

Figura 5. Archivo de proyecto HPJ.

BMP, WMF y SHG. Una vez abierto un gráfico sólo tenemos que pulsar el botón izquierdo del ratón y seleccionar un área del gráfico para crear un punto caliente (Hotspot), a continuación podemos definir los atributos del área seleccionada mediante la opción Atributes del menú Edit. Esta opción muestra un cuadro de dialogo que permite indicar el tipo de acción: Jump (salto a otro tema), Pop-up (desplegar una ventana) o Macro (ejecutar una macro).

USO DE MACROS

La ayuda de Windows posee un juego completo de macroinstrucciones que nos ayudaran a personalizar nuestro archivo de ayuda. Se puede dividir las 56 macros de la ayuda de Windows en las siguientes categorías: de botones, menú, enlace, ventana, teclado, auxiliares y de marcas de texto. La figura 4 muestra una breve descripción de cada macro, encontrará la lista completa en las SDK de Microsoft.

La sintaxis de una macro es la siguiente: Macro("parámetro1", "parámetro2", ...). Podemos ejecutar una macro al abrir el archivo de ayuda, al entrar en un tema, desde un botón o un menú o al hacer clic sobre el un texto o gráfico. Para ejecutar una macro al abrir el archivo de ayuda, la macro debe estar definida en la sección [CONFIG] del archivo de proyectos, como se muestra en la figura 5. Si insertamos el símbolo de nota al pie (!) en un tema, podemos indicar el nombre de una macro que se ejecutara cada vez que entremos en él, tal y como ocurre en el ejemplo que acompaña a este artículo, al entrar en el tema "Uso de Macros". La macro CreateButton nos permite insertar un botón con la siguiente sintaxis: CreateButton("id_botón", "texto", "macro"), donde id_botón es el identificador del botón, texto es el texto que se desea que muestre el botón y macro es la macro que se ejecutará al pulsar el botón. Con el subrayado doble de una o varias palabras o gráfico, inmediatamente después, y con el formato de texto oculto, el signo de admiración (!) junto al nombre de la macro, se ejecutará la macro al hacer clic sobre el texto o gráfico:

Macros Auxiliares

Back	Tema atrás.
BrowseButtons	Añade los botones "<<" y ">>".
ChangeButtonBinding	Cambia la función de un botón.
Contents	Presenta el índice de la ayuda.
CreateButton	Crea un botón nuevo.
DestroyButton	Destruye un botón.
DisableButton	Desactiva un botón.
EnableButton	Activa un botón.
History	Presenta la ventana de historial.
Next	Tema siguiente.
Prev	Tema anterior.
Search	Presenta el cuadro Buscar.
SetContents	Indica que tema es el índice.

Macros de teclado

AddAccelerator	Asigna una función a una tecla.
RemoveAccelerator	Anula la función de una tecla.

Macros de Enlace

JumpContents	Salta al índice de un archivo de ayuda.
JumpContext	Salta al tema especificado.
JumpHelpOn	Salta al Uso de la Ayuda.
JumpId	Salta al tema especificado por su identificador.
JumpKeyword	Salta al tema que contenga la palabra clave.
PopupContext	Abre una ventana pop-up del tema especificado.
PopupId	Abre una ventana pop-up del tema especificado por su identificador.

Macros de menú

About	Presenta el cuadro Acerca de la Ayuda.
Annotate	Presenta el cuadro Anotar.
AppendItem	Añade un menú.
BookmarkDefine	Presenta el cuadro Definir Marcador.
ChangeItemBinding	Cambia la función de una opción de menú.
CheckItem	Marca una opción de menú.
CopyDialog	Presenta el cuadro Copiar.
CopyTopic	Copia todo el texto al portapapeles.
DeleteItem	Borra una opción de menú.
DisableItem	Desactiva una opción de menú.
EnableItem	Activa una opción de menú.
Exit	Sale de la Ayuda.
FileOpen	Presenta el cuadro Abrir.
HelpOn	Presenta el Uso de la Ayuda.
InsertItem	Inserta una opción de menú.
InsertMenu	Añade un nuevo menú.
Print	Imprime el tema.
PrinterSetup	Presenta el cuadro Imprimir.
UncheckItem	Elimina una marca de una opción de menú.

Macros Auxiliares

ExecProgram	Ejecuta una aplicación.
RegisterRoutine	Registra una macro de una DLL.

Macros de Marcas de Texto

DeleteMark	Borra una marca.
GotoMark	Salta a una marca.
IfThen	Salto condicional a una marca.
IfThenElse	Salto condicional a una marca.
IsMark	Comprueba que una marca existe.
Not	Invierte el resultado de IsMark.
SaveMark	Guarda una marca en el tema actual.

Macros de Ventana

CloseWindow	Cierra una ventana secundaria.
FocusWindow	Cambia el foco de la ventana.
PositionWindow	Establece el tamaño y la posición de la ventana de la ayuda.

Figura 4. Lista de macros del Sistema de Ayuda de Windows.

[ALIAS] Sección	Asigna alias a los Identificadores.
[BAGGAGE] Sección	Lista de ficheros a incluir en el archivo de ayuda.
[BITMAPS] Sección	Especifica el directorio de los archivos gráficos.
BMROOT Opción de OPTIONS	Directorio de los gráficos.
BUILD Opción de OPTIONS	Temas a incluir en la ayuda.
[BUILDTAGS] Sección	Especifica los build tags
CITATION Opción de OPTIONS	Inserta una cadena en el cuadro Copiar.
COMPRESS Opción de OPTIONS	Indica el nivel de compresión del archivo de ayuda.
[CONFIG] Sección	Opciones de configuración.
CONTENTS Opción de OPTIONS	Especifica el índice de la ayuda.
COPYRIGHT Opción de OPTIONS	Inserta una cadena en el cuadro Acerca de..
ERRORLOG Opción de OPTIONS	Especifica el archivo que recoge los errores.
[FILES] Sección	Lista de los ficheros RTF.
FORCEFONT Opción de OPTIONS	Establece la fuente para el texto.
ICON Opción de OPTIONS	Especifica el icono del archivo de ayuda.
LANGUAGE Opción de OPTIONS	Establece el lenguaje para el orden de las palabras.
[MAP] Sección	Asocia en Identificador con un valor numérico.
MAPFONTSIZE Opción de OPTIONS	Establece el tamaño de las fuentes para el texto.
MULTIKEY Opción de	Especifica notas al pie alternativas.
OLDKEYPHRASE Opción de OPTIONS	Especifica si usar el archivo antiguo de frases.
OPTCDROM Opción de OPTIONS	Optimiza el archivo de ayuda para CD-ROM.
[OPTIONS] Sección	Opciones de compilación.
REPORT Opción de OPTIONS	Presenta mensajes durante la compilación.
ROOT Opción de OPTIONS	Especifica el directorio de los archivos RTF.
TITLE Opción de OPTIONS	Titulo para la ventana de ayuda.
WARNING Opción de OPTIONS	Especifica el nivel para los mensajes de error.
[WINDOWS] Sección	Configuración de la ventana de la ayuda.

Figura 6. Secciones del archivo de proyecto (HPJ).

texto!Macro( ), {bmc imagen.bmp}!Macro( ).

La macro RegisterRoutine permite registrar macros definidas por el usuario, que deben residir en una Librería de Enlace Dinámico (DLL). Podemos crear nuestras propias Librerías de Enlace Dinámico, como se vera en una próxima entrega, o hacer uso de las ya existentes. La sintaxis de esta macro es la siguiente: RegisterRoutine("nombreDLL", "nombreFunción", "formato"), donde nombreDLL es el nombre de la librería,

Podemos definir nuestras propias macros, creando una DLL, o hacer uso de las ya existentes

ésta debe residir en el directorio de windows, windows\system o en el mismo directorio del archivo de ayuda, nombreFunción es el nombre de la función en la DLL y formato es el formato de los parámetros con las siguientes equivalencias al lenguaje C: "u" unsigned short (WORD), "U" unsigned long (DWORD) , "i" short int, "I" int, "s" near char * (PSTR), "S" far char * (LPSTR) y "v" void. La macro RegisterRouite sólo puede estar en la sección [CONFIG] del archivo de proyectos. Una vez registrada la macro podemos utilizarla como si de una macro más se tratase. Para el ejemplo que acompaña a este artículo se ha registrado la función del API de Windows MessageBeep del USER.EXE, que emite un sonido, como se puede ver en la figura 5.

MÁS SOBRE EL ARCHIVO DE PROYECTOS HPJ

Además de las secciones [FILES] y [MAP], discutidas en la entrega anterior, el archivo de proyectos puede contener las siguientes: OPTIONS, donde se especifican las opciones de compilación, esta sección debe ser la primera del archivo de proyectos, CONFIG, donde se especifican las macros definidas por el usuario, botones, menús, etc. BITMAP, donde se incluye la lista de gráficos, no es necesaria si se ha definido la opción BMROOT de la sección OPTIONS, o si los gráficos residen en el mismo directorio del archivo de proyectos, WINDOWS, donde se definen las características de la ventana de la ayuda. La figura 6 muestra la lista de opciones del archivo de proyecto, encontrará la lista completa en las SDK de Microsoft.

Nota al pie	Significado
#	Identificador del Tema (Context string)
$	Titulo, aparece en cuadro Buscar e Historial
K	Palabras clave que aparecerán en el cuadro Buscar
+	Browse sequence, determina el orden para los botones "<<" y ">>"
*	Build tag, identificador que permite incluir o no un tema en la ayuda
!	Macro, ejecuta una macro

Figura 7. Notas al pie.

Para el ejemplo que acompaña este artículo se ha definido en la sección [OPTIONS] la opción COPYRIGHT que inserta una cadena de no más de 50 caracteres en el cuadro Acerca de la Ayuda, la opción TITLE que permite especificar un titulo para la ventana de la ayuda, la opción COMPRESS que permite tres niveles de compresión para el archivo de ayuda, FALSE sin compresión, MEDIUM, y HIGH para el máximo. Es posible que al utilizar opción COMPRESS, o si el archivo de ayuda contiene muchos o gráficos muy grandes, se visualice un mensaje de error al compilar, para solucionar este problema hay que utilizar el compilador HC31P.EXE, que utiliza memoria extendida, en lugar de HC31.EXE. Otro inconveniente de la opción COMPRESS es que relentiza el proceso de compilación, por lo que es conveniente activarla únicamente cuando se va a crear el archivo de ayuda definitivo, y tenerla desactivada durante la depuración. También es posible que nos encontremos con problemas en los acentos, para evitar esto edite el archivo de proyectos con el editor Edit del DOS.

CONCEPTOS BÁSICOS

Daniel Navarro



Toda traducción es siempre problemática, dado que es difícil en muchas ocasiones expresar la misma solución a un problema en lenguajes de naturalezas tan diferentes como C o ensamblador. Esa es la misión de nuestra, por excelencia, herramienta de trabajo. Es tan vital conocer bien el compilador que utilizamos, como conocer bien el lenguaje que implementa.

El corazón de un compilador es un traductor, el cual es un programa que recibe como entrada un algoritmo expresado en un lenguaje y emite como salida el mismo algoritmo expresado en un lenguaje diferente. Un compilador es un paquete que incluye muchas más cosas aparte del traductor, todas ellas necesarias para programar, como son: librerías de funciones, entorno de desarrollo, depurador o trazador de código, ayuda en forma de documentación, un optimizador de código, información sobre los errores, etc.

Esta serie de artículos, que se debería llamar 'TRADUCTORES', se centra en el interior de los compiladores, con un primer objetivo que es mejorar el uso que hacemos de nuestra herramienta de trabajo, aumentando de forma considerable nuestra 'cultura' informática. El segundo objetivo es enseñar lo justo y necesario para aquellos de nosotros que sintamos la necesidad de inventar un nuevo lenguaje de programación que revolucione el mundo.

Desmitificar los compiladores será la primera meta, nadie va a hundir a Microsoft o Borland inventando el D++ o el C orientado a seres vivos, pero inventar un lenguaje y programar un traductor para él es algo que cualquiera de nosotros puede hacer. Creame que si lo hacen, comprobar como funciona un pequeño programa en su nuevo lenguaje será una satisfacción que merecerá la pena con creces, y habrás aprendido muchas cosas útiles en el camino.

¿Por qué inventar un nuevo lenguaje y para qué? todos los tipos de problemas tienen un lenguaje ideal para plantear su solución. Un lenguaje para manejar bases de datos, otro para problemas de física o matemáticas, otro para programar multimedia, otro para programar equipos electrónicos y otro para programar vídeo juegos.

CONCEPTOS

Antes de entrar en materia, es necesario aclarar muchos conceptos y vocabulario del mundillo de los compiladores.

TRADUCTOR: como se mencionó antes se trata de un programa que recibe y devuelve el mismo algoritmo, pero expresado en dos lenguajes diferentes. Al lenguaje que recibe se le denomina lenguaje fuente, y al que devuelve lenguaje destino. Los traductores más genéricos que usamos nosotros son aquellos que traducen programas escritos en un lenguaje de alto nivel (C, C++, Pascal, ...) a código máquina para los procesadores Intel 80x86, pero también podemos hacer un traductor de Pascal a C++, que tome un programa en el primer lengua-

Usted que es un programador, ¿cómo definiría un compilador? como un amigo o como un enemigo. La utilidad de un lenguaje de programación es ayudar al programador a expresar las soluciones a los problemas que debe resolver, y la del compilador traducir la solución encontrada para que la entienda el tonto de nuestro ordenador.

je y produzca una salida válida para ser compilada con un compilador comercial de C++. Un traductor también puede traducir de ensamblador para z80 a ensamblador para 68000, o recibir un programa en ensamblador 386 y producir como salida un programa equivalente en Qbasic (aunque con sus limitaciones, obviamente).

INTÉRPRETE: los intérpretes son programas que, por decirlo de alguna manera, juegan a ser microprocesadores. Leen un programa y van realizando secuencialmente lo que el programa expresa, simulan su ejecución, pero en realidad lo que siempre se está ejecutando es el código del intérprete. Su desventaja es que este proceso es más lento que generar código máquina y dejar que el procesador de nuestro ordenador lo 'interprete'. Su principal ventaja es que son muy sencillos y flexibles, permitiendo un control total sobre la ejecución. Es más fácil interpretar un programa en Basic que generar código máquina equivalente (con un programa traductor).

LENGUAJE: todos sabemos que es un lenguaje de programación, pero este concepto abarca mucho más allá de lo que ahora imaginamos. Un lenguaje de programación, además de Módula, C++ o Cobol, puede ser por ejemplo el binario.

Es muy fácil hacer un traductor de programas escritos en binario a código maquina (cogemos ocho caracteres del programa fuente que queremos traducir, si los ocho son caracteres '0' o '1' escribimos el byte equivalente a esa combinación binaria en el fichero destino, repitiendo este proceso hasta que se terminara el programa fuente, y si viniera un carácter diferente emitimos un error de compilación '**Carácter no válido**), muy fácil sí, pero a ver quién es el valiente que se programa un simulador de vuelo en ese lenguaje. Como otro ejemplo, podríamos inventarnos un lenguaje para programar una caldera de calefacción, este podría ser una lista de instrucciones de dos tipos, una "GOA x" indicaría que queremos que la caldera suba o baje hasta la temperatura indicada en "x" y otra "WAT x" indicaría que se mantuviera en el estado actual durante "x" horas, el programa para un día podría ser:

GOA 18 WAT 2 GOA 20 WAT 6 GOA 16 WAT 2 GOA 0 WAT 14

Y si además permitiéramos que hubiera un salto de línea entre instrucción e instrucción, entonces hasta parecería un programa de verdad.

NIVEL DE UN LENGUAJE, Se habla siempre de lenguajes de alto o bajo nivel, esto se refiere simplemente a si están más cerca del nivel de comprensión humana (alto nivel) o del nivel de comprensión de la máquina (bajo nivel). No se refiere nunca ni a la calidad ni a la complejidad de los lenguajes, así por ejemplo, aunque los dos lenguajes que utilizamos en el punto anterior son muy sencillos, el lenguaje binario para programar un simulador de vuelo sería de muy bajo nivel, mientras que el lenguaje de la caldera sería de alto nivel, ya que está muy cerca de como pensamos nosotros (primero tantas horas a tal temperatura, después baja un poco ...), aunque un lenguaje para programar la caldera con un nivel mucho más alto sería:

BEGIN
Por favor, si no te importa, enciéndete a las 10 de la mañana y pon una temperatura agradable, calentita pero sin pasarte, después si calienta mucho el sol me bajas un pelín, ¿vale?.
END

Como se observa, ni siquiera la longitud de la solución a un problema depende de que el lenguaje que se empleó para resolverlo fuera de un nivel más bajo o más alto.

SÍMBOLO: cada lenguaje tiene sus elementos bien definidos. Un lenguaje como el castellano tiene palabras. En los lenguajes formales a estos elementos no se les denomina palabras, sino símbolos, son símbolos del lenguaje C, por ejemplo, tanto la palabra 'main' como los símbolos '{','+' o ';' o una constante numérica.

Son todos elementos con identidad propia, y con un significado y una utilidad determinadas, no son símbolos de C, por ejemplo, ni los espacios en blanco ni los saltos de línea, pues no intervienen en absoluto en la estructura del programa.

Un programa siempre se considera como una secuencia de símbolos del lenguaje. Por ejemplo, si tenemos el siguiente programa escrito en C:

main() { / Comentario */*
int número=28; }

Lo primero que haría un traductor es leer la secuencia de caracteres que conforma el programa fuente (' ','m','a','i','n','(', ...) e identificar los símbolos del lenguaje, después de hacerlo el programa anterior sería para un traductor algo como esto:

símbolo (main), símbolo (abrir paréntesis), símbolo (cerrar paréntesis), símbolo (inicio bloque), símbolo (int), sím-

```
      Dis_Mio[iDis_Mio]=X_Nave+1
      iDis_Mio=(iDis_Mio+2)%48
   fin

   Si tecla="2" y Niv_Actual<14*13
      Niv_Actual=Niv_Actual+13 Ir.Inicio_Nivel fin
   Si tecla="1" y Niv_Actual>0
      Niv_Actual=Niv_Actual-13 Ir.Inicio_Nivel fin

   Hacer.Pinta_nave
   Hacer.Control_Enemigos
   Hacer.Control_Disparos Hacer.Control_Disparos
   Hacer.Control_Bombas

   Si Muerto
      Mover_cursor(X_Nave+1,24)
      Color(78) Escribir("")
      Color(15) Escribir("")
      Color(78) Escribir("")
   Fin

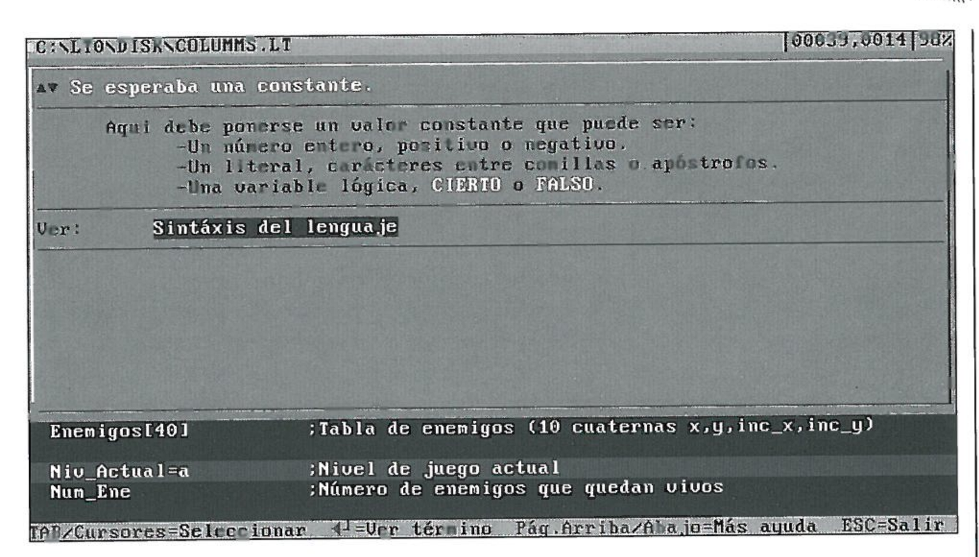
   Esperar(Niveles[Niv_actual+9])
Hasta tecla="q" o tecla="Q" o Num_Ene=0 o Muerto

Si Muerto
   Si Vidas>=0 Ir.Inicio_Nivel ;Se perdió una vida
   Sino Hacer.Cls2
        Color(15) Mover_cursor(35,12)
        Escribir("GAME OVER")
        Hacer.Pausa
        Borrar_pantalla() Ir.Menu ;GAME OVER
   Fin
Fin

Si Num_Ene=0
   Si Niv_Actual<14*13 Niv_Actual=Niv_Actual+13 Ir.Inicio_Nivel ;Nivel+1
   Sino ;Se llegó al final
   Fin
Fin

hacer.cls Borrar_pantalla() Ir.Menu ;QUIT - Se abortó la partida
```





bolo (identificador de una variable), símbolo (asignación), símbolo (constante numérica), símbolo (separador de sentencia), símbolo (fin de bloque)

Y para poder 'trocear' de esta manera el programa fuente, el traductor debe aplicar las reglas léxicas.

REGLAS LÉXICAS, son un conjunto de normas que definen la forma que tienen estos elementos básicos del lenguaje, por ejemplo, se puede especificar que un símbolo de tipo identificador es una secuencia de caracteres que comienza con un carácter alfabético (entre 'A' y 'Z' mayúsculas o minúsculas), y continúa mientras sigan en secuencia caracteres alfanuméricos (letras o números). Si los definimos así un traductor analizaría "Mivar2" como un símbolo identificador, pero "Mi_var2" serían dos símbolos identificadores con otro símbolo extraño entre ellos.

También podemos especificar que un símbolo de asignación es la secuencia de caracteres ":=" (dos puntos y un igual). Con estas dos sencillas reglas ya hemos definido dos símbolos del lenguaje, los identificadores y los de asignación.

En un lenguaje de programación inventado que tuviera solo esos dos símbolos no sería correcto léxicamente cualquier símbolo que no estuviera englobado en una de esas dos definiciones (como '+', '8a', '==' o '_'), y cuando un programa traductor los recibiera como entrada emitiría un 'error lexicográfico', para advertirnos que hemos violado las reglas léxicas del lenguaje que implementa.

REGLAS SINTÁCTICAS, una vez definidos cuales son los símbolos de un lenguaje, con las reglas sintácticas (o de sintaxis) se definen que secuencias de símbolos son válidas dentro del lenguaje. Por ejemplo, en el lenguaje binario anteriormente aludido, sólo hay dos símbolos posibles, el cero y el uno, y sus reglas sintácticas son bien sencillas: "un programa es cualquier secuencia de símbolos". Pero el lenguaje binario es un caso especial (dada su extrema simplicidad), pues lo más normal en todos los lenguajes de programación es que solo sean válidas algunas de las posibles secuencias de símbolos; en Pascal por ejemplo, es válida la siguiente secuencia:

```
Ancho := Alto + 1 ;
```

No siendo válida (no perteneciendo al lenguaje) esta otra secuencia:

```
Ancho := ; + Alto 1
```

El traductor detectaría que la secuencia no es válida porque en alguna de sus reglas sintácticas se especifica que: "Después de un símbolo de asignación (:=) no puede venir un símbolo separador de sentencias (;)". Es posible que una regla como la anterior no esté especificada explícitamente, pero sea deducible del resto de reglas especificadas en el lenguaje.

De una secuencia de símbolos correctamente construida se dice que es una secuencia correcta sintácticamente, y cuando un programa no cumple estas normas (como en el caso del último ejemplo), el programa traductor emite un error sintáctico (Syntax error) informando que no respetamos sus reglas sintácticas.

REGLAS SEMÁNTICAS: son las últimas normas que se imponen para que un programa se pueda dar por correcto. Son las más genéricas y variables de un lenguaje a otro. Una vez tenemos un programa que cumple las reglas lexicográficas (que tiene los símbolos bien formados) y las sintácticas (bien concatenados), hemos de verificar que esta bien construido. Veamos el siguiente programa en C:

```
main(){
  n=33;
  printf("El valor de n es %u",n);
}
```

Este programa es correcto léxicamente (no hay símbolos mal construidos) y sintácticamente (tampoco hay problemas con la secuencia de símbolos), pero no es correcto por que no hemos declarado la variable 'n', y debería haber aparecido antes una declaración del tipo: "int n;". Esta situación se denomina error semántico.

Las reglas semánticas definen cosas como todo lo referente a la coherencia de las expresiones, dicen si es posible operar con variables de tipos diferentes, que el número de parámetros en una llamada a una función debe coincidir con el número de parámetros que esta recibe (y coincidir los tipos), que se deben declarar todos los recursos (como constantes, procedimientos, variables, tablas, ...) que se utilicen en el programa, que una variable no se puede declarar varias veces, etc... Son el tipo de normas que definen si un programa es correcto.

LO QUE EL COMPILADOR NO COMPRUEBA

Cuando un programa infringe alguna de las normas del lenguaje (léxica, sintáctica o semántica) el traductor no puede traducir correctamente el programa,

pues no está escrito en el lenguaje que el conoce. No obstante, que el programa cumpla todas las reglas del lenguaje no garantiza que el programa sea correcto, aún quedan los errores de concepto, y estos son errores que el traductor no puede detectar. Por ejemplo, el siguiente programa para calcular la media de dos números es correcto para un compilador de Basic.

10 INPUT A,B
20 PRINT (A+B)/38

Pero, evidentemente, no es un programa correcto, pues no consigue su propósito que era calcular la media de dos números, tiene un error de concepto (dividir por 38 y no por 2).

FASES DE UN TRADUCTOR

Un programa traductor divide su trabajo en dos partes, el análisis y la síntesis.

La fase de análisis consiste en recibir un fichero como entrada el programa fuente y analizándolo secuencialmente comprobar la estructura del programa, viendo si se cumplen todas las reglas del lenguaje (léxicas, sintácticas y semánticas). Dependiendo de si las cumple o no, el traductor decide si 'ha entendido' el programa o 'no lo ha entendido'. Si lo ha entendido entonces puede pasar a la síntesis, en caso contrario se produce un error de compilación y posiblemente no se pueda pasar a la siguiente fase (a no ser que el compilador pueda 'remendar' el error de alguna manera y seguir compilando).

La fase de síntesis es la que 'explica' en el lenguaje destino lo que decía el programa en el lenguaje fuente. ¿Y cómo lo explica?, pues sencillamente generando él otro programa en el lenguaje destino, que hace lo que entendió que quería hacer el programa fuente.

LAS PASADAS SOBRE EL FUENTE

Un programa traductor puede necesitar leer varias veces el programa fuente para entender lo que en este se especifica. La forma de funcionar un traductor en general, es ir leyendo y analizando todas las líneas del programa por orden, primero la línea uno, luego la dos, etc... y hay algunos problemas que para resolverlos, el traductor necesita hacer un segundo (tercer, o cuarto) análisis de las líneas del programa. Un ejemplo: vamos a suponer que tenemos que traducir un programa escrito en un supuesto lenguaje que tiene etiquetas y saltos incondicionales a esas etiquetas (GOTO o JMP) ¿Que hace el traductor cuando se encuentra un salto a una etiqueta que no conoce y no sabe donde está?.

Emitir un error no es una solución, pues no permitiría saltar a etiquetas situadas al final del programa desde su inicio, ya que cuando llega la instrucción de salto no ha sido encontrada todavía la etiqueta correspondiente. Una primera solución para poder tratar correctamente los saltos, sería leer dos veces el fuente, la primera nos quedamos con la posición de las etiquetas y en la segunda pasada (o lectura del fuente) trataríamos correctamente las instrucciones de salto (generando un salto a la dirección en la que estuviera definida la etiqueta). Y una segunda solución sería leer sólo una vez el fuente (una sola pasada) anotando en variables del traductor todas aquellas etiquetas que han sido utilizadas y aún no han sido declaradas.

Como hemos visto, según sea diseñado un traductor puede ser de una o múltiples pasadas. En general es más complicado hacerlo todo en una misma pasada, pero es mucho más rápido que hacerlo en varias. Hay lenguajes que por su diseño se dice que son de una o de varias pasadas, esto se refiere a si al implementar su traductor es suficiente hacer una sola pasada o es necesario hacer varias. Hay muchos más motivos que la complejidad del lenguaje para hacer más o menos pasadas, los compiladores suelen hacer pasadas extra cuando se pretende optimizar el código, por ejemplo.

Es importante no confundir las pasadas que hace un traductor sobre un programa con las fases del compilador, por ejemplo, se pueden hacer varias pasadas para la fase de análisis o bien hacer en una pasada la fase de análisis y parte de la fase de síntesis, requiriendo otra pasada para finalizar esta segunda fase.

TRATAMIENTO DE LOS ERRORES

Durante la fase de análisis, que es donde se lee el programa fuente y se 'entiende' (comprobando además todas las reglas del lenguaje), el compilador realiza una de sus funciones más importantes (y más complicadas) que es tratar los errores que se produzcan de la mejor manera que sepa.

Hay una solución bastante fácil que consiste en leer el programa fuente secuencialmente comprobando todas las reglas (primero las léxicas, después las sintácticas y por último las semánticas), y si en un momento dado se incumple alguna de ellas se informa de qué regla se incumplió en que línea del programa, abortando el proceso de traducción. Pero esto no es lo que hacen los compiladores, ¿conoces alguno que no sea capaz de encontrar varios (y no sólo uno) errores de compilación en una sola ejecución?, los hay pero esta no es la solución más profesional, ya que si estamos utilizando un compilador externo que tarda 2 minutos en compilar nuestro programa y tenemos que ejecutarlo para ver que nos hemos olvidado declarar la variable 'Contador', arreglarlo, volver a compilar para ver que también se nos olvidó un punto y coma, arreglarlo, ... cuando podría habernos informado de todos los errores en la primera compilación, posiblemente nos acordemos de la familia del 'que-se-llama-programador' y tuvo la feliz idea de "si con un solo error vas que te matas ...".

Pero pensemos un momento, ¿qué tiene que hacer entonces un compilador cuando se produce un error?, Si escribimos en nuestro programa (por ejemplo en Pascal):

```
Mi_jugada_es := mejor que la_tuya;
```

El compilador no puede saber que es lo que pretendíamos escribir: si es



una sentencia, si son varias y se nos olvido separarlas por puntos y coma, si es un comentario que se nos olvido iniciar, etc... ¿Y qué hacen los compiladores en este caso? Pues informarnos, no de uno, sino de cuatrocientos veintisiete errores de compilación, ¿por qué? Pues porque no puede adivinar cúal es el error que realmente hemos cometido, y si quiere seguir compilando para poder detectar más errores en la misma ejecución debe hacer una suposición. Pero es realmente muy difícil hacer una suposición y que esta sea correcta, quizá lo más fácil es suponer que toda la parte errónea era un comentario, y olvidarse de ella, pero si cometimos un error al declarar una variable y el compilador supone que esa declaración es un comentario sin importancia, se producirán muchos errores por todo el programa cada vez que utilizemos esa variable que, para el compilador, no ha sido declarada. Aún así, las cosas no son tan sencillas, ya que cuando se produce un error debemos decidir cuanto código vamos a ignorar cerca del error, y decidir también cuando el código vuelve a tener sentido tras el error. Los lenguajes de alto nivel modernos suelen tener un símbolo para separar sentencias, y su principal cometido es precisamente este, cuando se produce un error, se supone que la sentencia actual no es válida y se ignora todo lo que venga hasta el siguiente símbolo separador (que en Pascal y C es un punto y coma).

GENERACIÓN DE CÓDIGO

Si se piensa que lo más complicado de un compilador no es el tema de los errores, ni el comprobar si se cumplen o no todas las reglas, sino que es el generar código destino, se está en lo cierto.

Un compilador, en la fase de síntesis, con la información sobre el programa ya estructurada, suele dividir el trabajo de generar código destino en varias partes:

En la primera ya genera todo el código, pero no el definitivo, es decir, se traduce todo el programa completo de golpe, pero no se traduce directamente al código destino, sino que se pasa a un código intermedio, similar al código máquina, aunque más sencillo y potente, que por decirlo así, está a medio camino entre el lenguaje fuente que teníamos que traducir y el lenguaje destino que tenemos que generar. Algunos de los motivos para hacer esto son: es más fácil generar este código que el código destino directamente (pues se diseña con ese fin), y se puede optimizar más fácilmente este código que el código destino.

Aunque quizá sea algo pronto, podríamos ver un ejemplo para poder imaginarnos de forma gráfica este proceso. Supongamos que un traductor debe pasar de C a ensamblador la siguiente sentencia:

```
Alfa=Alfa+1+Beta;
```

Primero en la fase de análisis obtendrá la información necesaria sobre las variables; algo como esto:

"Alfa", de tipo entero sin signo (2 bytes); se ha ubicado en la dirección 23002.

"Beta", de tipo entero sin signo (2 bytes); se ha ubicado en la dirección 23004.

Y entonces, cuando en la fase de síntesis genere código para la anterior sentencia, generará un código (en alguna tabla interna del compilador) similar al siguiente:

```
CARGA (23002)
CARGA 1
SUMA
CARGA (23004)
SUMA
DESCARGA (23002)
```

Una vez traducido todo el programa a este código, continuará la fase de síntesis. Para pasar de este código intermedio a el código destino ya no se volverá a mirar el programa fuente. Si se tuviera que pasar el programa a código máquina Intel 80x86, el resultado de la sentencia anterior podría quedar de la siguiente manera:

```
INC (23002)
MOV AX,(23004)
ADD (23002),AX
```

Pero este resultado depende de los algoritmos utilizados para generarlo (gestión de los registros, optimizaciones, etc.)

PRÓXIMO NÚMERO

Esto es todo lo que se necesita para poder estudiar a partir del siguiente número como se construye un compilador. Empezaremos con el diseño del lenguaje y el reconocimiento de los símbolos, para en muy poco tiempo tener nuestro compilador funcionando. Aquellos que esten interesados pueden encontrar en el cuadro "Un ejemplo práctico", información sobre un compilador de un lenguaje inventado, que también se incluye en el disco de este mes. En esta serie vamos a desarrollar este compilador (con alguna pequeña innovación) ... y vosotros iréis creando el vuestro.

UN EJEMPLO PRÁCTICO

Este mes el lector podrá observar un mini-compilador para un mini-lenguaje, muy similar al que vamos a desarrollar en esta serie.

El lenguaje que nos hemos inventado se llama "Letra" y es un lenguaje de propósito general en castellano de naturaleza muy similar a otros lenguajes que podamos estar manejando.

El compilador es el fichero LT.EXE y los programas fuentes tienen extensión "*.LT", podemos compilar cualquiera de los programas ejemplo (Tecleando en la línea de comandos, por ejemplo: LT TABLAS) y el compilador generará un ejecutable (En el ejemplo: TABLAS.COM).

Sin embargo es recomendable usar el compilador desde el entorno de programación (cargando E.EXE y pasándole como parámetro uno de los programas), desde el entorno podemos modificar los programas o crear otros nuevos. Es muy fácil programar en este lenguaje, sólo con examinar alguno de los ejemplos (como TABLAS o COLUMMS) todas las dudas que os asalten se podrán resolverlas con la ayuda (disponible en cualquier momento pulsando F1). También se puede cambiar la resolución con F8, grabar los programas con F2, compilarlos con F5 y ejecutarlos con F9. Para abortar la ejecución de cualquier programa (incluso si se ha quedado bloqueado) o para salir del editor basta con pulsar la tecla ESC.

CONVERSIÓN DE SONIDO PCM A SPEAKER

Enrique de Alarcón

En la página 71 del número 11 de Sólo Programadores, apareció un artículo sobre la programación del temporizador 8253/8254. En el mismo , se daba como ejemplo de sus aplicaciones la salida del sonido digital a través del altavoz interno (speaker) , pero no se detalló , por quedar fuera de su tema central, el método para convertir las señales PCM a frecuencias equivalentes generables por el altavoz interno del PC. A continuación , se explica todo lo relacionado con ello , tanto el funcionamiento de las ondas PCM , formato estándar de sonido digital de la Creative Labs , creadora de toda la familia de tarjetas de sonido Sound Blaster , como el funcionamiento del altavoz interno y de la conversión PCM/Speaker y sus limitaciones.

EL SONIDO DIGITAL

Cuando digitalizamos un sonido a 11000Hz mono en 8 bits en PCM (Pulse Code Modulation), por ejemplo , en una Sound Blaster , lo que estamos haciendo es almacenar los valores de intensidad de sonido 11000 veces por segundo (o sea tomando una muestra de sonido cada 0.000091segundos).
Cada instante capturado (cada muestra) , será 1 byte (en el caso de grabar a 8 bits) con valores comprendidos entre 0 y 255 y el cual corresponderá al nivel de intensidad.

La intensidad de volumen 0(sonido nulo) , equivale al valor 128 , el valor 0 equivale a la intensidad mínima , y el valor 255 corresponde a la intensidad máxima.

Gráficamente , podríamos dibujar la onda de sonido dando los valores de intensidad a la Y y la evolución del tiempo en la X.

En la misma , la Y=0, correspondería al valor digital 128 , que como hemos dicho , corresponde a sonido nulo , la Y=128 , sería la intensidad máxima , o sea 255 y la Y=-128 corresponde al valor digital 0.

Como se puede observar en el ejemplo de la figura 1 (cualquier editor de sonido WAV de windows o similar permite ver este tipo de gráficas) las señales positivas y negativas se suceden continuamente (prácticamente se van alternando) y la gráfica resultante parece dar dos dibujos iguales y simétricos en cada una de las zonas positivas y negativas de la gráfica.

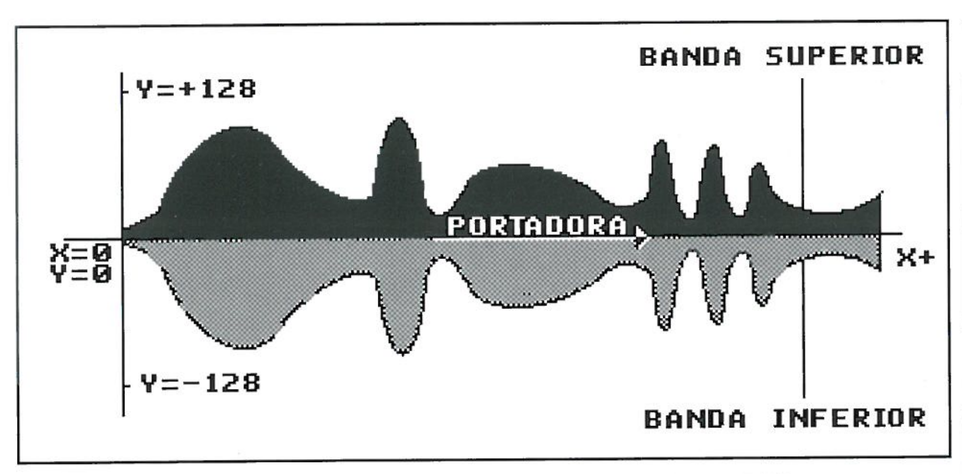
Técnicamente , el nivel 0 corresponde a la <señal portadora> y cada una de las zonas positivas y negativas del dibujo corresponden a las <bandas laterales> de la señal.

PCM son las siglas de Pulse Code Modulation

Saber convertir sonido digitalizado de formato PCM a frecuencias de altavoz interno puede darnos la oportunidad de reproducir sonido en ordenadores que no dispongan de tarjeta de sonido así como un mayor conocimiento sobre la manera de funcionar del mismo para poder de esta forma manipularlo a nuestro gusto.

Para comprender mejor el funcionamiento , damos dos ejemplos. Si se grabara un sonido nulo puro (sin ningún tipo de ruido de fondo) , la gráfica resultante sería una línea recta con la Y siempre con el valor 0 a lo largo del tiempo , o sea , un valor 128 en cada byte de sonido capturado.

Otro ejemplo, en un sonido bastante alto , la gráfica contendría valores cercanos a +128Y y -128Y alternándose continuamente , que sería lo mismo que decir que las muestras de



<Ejemplo de representación gráfica de las muestras de un sonido en PCM>.

sonido tomadas tenían intensidades de sonido altas.

FUNCIONAMIENTO DEL ALTAVOZ INTERNO

La generación de sonido en el altavoz interno , se realiza enviando una señal binaria 1 seguida de un valor 0 al mismo, lo cual da un 'clic' de sonido.

Por lo tanto , para generar una frecuencia de 6000Hz , corresponde a enviar 6000 señales binarias 1 con sus respectivos 0 posteriores dentro del transcurso de 1 segundo , lo cual dará 6000 vibraciones en 1 segundo = 6000Hz.

A este modo de funcionamiento , se le llama producir ondas de tipo cuadrado , ya que si dibujásemos una gráfica , sólo podríamos escribir niveles de intensidad 1 y 0 sin niveles intermedios de ascenso ni descenso de las ondas.

Como esto se hace mediante el temporizador (Contador 3) y como en este puede programarse periodos de entre 0 y 65535 (un número de 16bits) y el reloj del sistema funciona a 1193180 sólo pueden obtenerse frecuencias de entre 18.2Hz (1193180/65535) y 1.193Mhz (1193180/1).

CONVERSIÓN DE SEÑALES PCM A SPEAKER

Convertir señales digitales PCM es sólo crear una equivalencia entre intensidades de volumen y frecuencias normales que sean generables por el <Speaker>. Para ello , debemos conocer antes algunos detalles técnicos:

La señal portadora , siempre equivale a un 66% de la frecuencia de muestreo , y cada banda (superior + , e inferior -) posee un 17% restante y lo cual suma un total de 100% de la frecuencia. En el caso del ejemplo puesto hasta ahora , que era 11000. sería:

Frec.Portadora= ((11000*66)/100) = 7260Hz.

Banda Superior=((11000*17)/100) = 1837Hz.

Banda inferior= ((11000*17)/100)= 1837Hz.

La generación de sonido en el altavoz interno , se realiza enviando una señal binaria 1 seguida de otra de valor 0

También debemos saber que para la reproducción por el altavoz interno , debemos eliminar una de las dos bandas y dejar sólo una a la que le corresponderá un 17%+17% de frecuencia total , del margen de frecuencias. Normalmente , eliminamos la banda que corresponde a las bajas frecuencias , por lo tanto , todas las muestras que tengan valores menores a 128 , las dejamos a 0 , y las que estén por encima de 128 , les restamos 128 (considerando bytes sin signo) quedando de esta forma sólo valores de intensidad positivos de 0 a 127.

Si el sonido estaba grabado en estéreo , sólo usaremos las muestras correspondientes a uno de los dos canales , sin importar si es el izquierda o el derecho , para ello , debemos ignorar las muestras impares , para eliminar el derecho , o las pares , para eliminar el izquierdo.

Una vez tenemos esto , hay que conseguir que cada incremento de intensidad de los 128 niveles posibles , sea equivalente a una fracción proporcional dentro del rango da frecuencias que tenemos por banda (34% de 11000 es 3740Hz).

Para esto , sólo debemos dividir el 34% de la frecuencia , que es el rango de frecuencias libre que nos queda del total , entre el número de intensidades:

Frec_Unidad= (3740/128) = 29hz.

Ya tenemos pues todos los datos. A cada muestra le corresponde una frecuencia Speaker de:

Frec.Portadora + (Frec_Unidad * Valor)

En el ejemplo sería:

Frecuencia Speaker= (7260 + (29*Valor))

El resumen para la conversión de un sonido sería como sigue:

```
Valor_Actual=Puntero a inicio del sonido;
Frec_Portadora= ((66*11000)/100);
Frec_U= ((11000 - Frec_Portadora) /
128);
 For(Longitud_Sonido);
  {
   M_Act= *Valor Actual;
   if (M_Act<128) M_Act=0;
     else M_Act-=128;
       *Valor_Act= (Frec_Portadora +
(Frec_U* M_Act));
                    Valor_Act++;
  }
```

SOBRE EL ORIGEN DEL SONIDO

Normalmente partimos de la base de que conocemos en que formato digital han sido grabados los sonidos digitalizados que vamos a usar en nuestras aplicaciones. Pero en el caso de usar ficheros de contenido técnico no conocido, debemos tener algunos datos presentes.

Si el sonido que vamos a convertir , lo extraemos de un fichero VOC (por ejemplo) , debemos saber de que este ,

al igual que muchos otros formatos de sonido, puede estar almacenado en formatos llamados <empaquetados> , sin contar ello que puede estar grabado a cualquier frecuencia de muestreo y tanto en estereo como en mono y tanto en 8 como en 16bits.

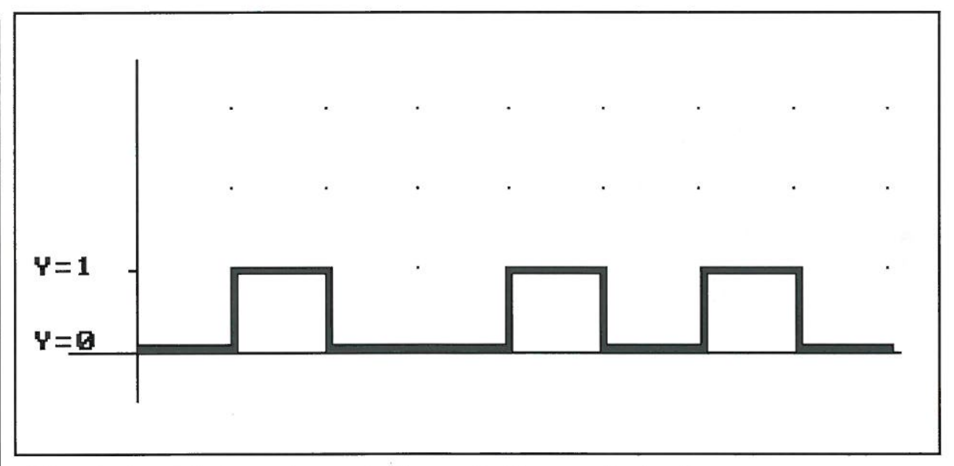
Así pues , para que tengamos una idea , en un fichero normal del tipo VOC , podemos encontrarnos hasta nueve tipos de formatos de digitalización diferentes:

*8-bits unsigned PCM.
*Creative 8-bits to 4-bits ADPCM.
*Creative 8-bits to 3-bits ADPCM.
*Creative 8-bits to 2-bits ADPCM.
*16-bits signed PCM.
*CCITT a-Law.
*CCITT u-Law.
*Creative 16-bits to 4-bits ADPCM.

En caso de que se trate del primer tipo o del quinto , no hay ningún problema , pues contiene muestras de sonido tal cual , pero en los demás casos , son sistemas de compresión que se soportan directamente por el hardware de las tarjetas Sound Blaster (por el DSP de la tarjeta) y que son algo complicados de convertir.

Los sistemas 8bits to Xbits ADPCM , son sistemas que se basan en tener la muestra inicial entera , y los valores posteriores , que son números de Xbits son sólo las diferencias de valor de intensidad respecto a la muestra anterior. De esta manera , por ejemplo , en el sistema de 8 a 4 bits , el ahorro de memoria es de un 50%. A pesar de que todo no es positivo ,dado que las grabaciones realizadas con estos sistemas no tienen la misma calidad que las normales.

De los sistemas CCITT a/u-Law , sólo sé que se trata de otro sistema de compresión de sonido parecido al ADPCM , pero no tengo referencias concretas al respecto. Las siglas CCITT provienen de <Comite Consultivo Internacional de Telegrafía y Telefonía> , un organismo responsable de elaborar las normativas para todos los temas relacionados con sus siglas.



<Ejemplo visual de las ondas cuadradas producidas por el <Speaker>.

ULTIMAS CONSIDERACIONES

Sabiendo como funciona el sonido PCM , podemos también realizar manipulaciones sobre el sonido que tengamos. Por ejemplo , y como fácilmente se puede deducir , para bajar el volumen del sonido , sólo es necesario reducir el rango de intensidades, por ejemplo , dividiendo todas las muestras entre 2 , obtendremos un sonido la mitad de fuerte. Y de la misma forma se pueden explicar muchos efectos , como ecos y otros...

Convertir señales digitales PCM es sólo crear una equivalencia entre intensidades de volumen y frecuencias normales

Un problema que se nos puede plantear , es convertir sonido digital de 16 bits a frecuencias speaker por ejemplo a 11000hz (la frecuencia más normal en reproducciones por altavoz interno). Como podemos deducir , los 3470Hz que tenemos para repartir entre los diferentes niveles de intensidad de sonido (en 11Khz) no son suficientes para incluir los 32767 niveles que permite la calidad de 16bits. En este caso , la única solución es perder <definición> de sonido dividiendo cada muestra de 16bits entre 10 , quedando así un rango de 3276 que cabe perfectamente dentro del rango de 3470Hz.

El tema de la frecuencia de muestreo también debe tenerse presente , pues no se puede reproducir , al menos en un ordenador actual , sonido por <Speaker> a 44Khz , o sea calidad CD , pues como se puede comprender , la CPU no aguantaría 44000 interrupciones de temporizador por segundo. En esta situación tan sólo tenemos que dividir la frecuencia de muestreo entre 4 (que dará el valor factible de 11Khz) y reproducir sólo las muestras múltiples de 4.

Otro dato importante , y que muchas veces no se tiene en cuenta es que la calidad de sonido que vamos a oír por el <Speaker> depende tanto de la calidad de grabación como de la calidad del propio altavoz , ya que se dan casos de ordenadores que poseen un <zumbador> (Speaker) muy malo , y el programador atribuye la mala calidad de sonido a la rutina de conversión de frecuencias o a la calidad de grabado , siendo en realidad de un problema físico.

También es cierto que por bueno que sea el altavoz , nunca se conseguirá una calidad que se pueda comparar ni de lejos , con la que se consigue con una tarjeta de sonido y que se trata sólo de intentar paliar parcialmente la falta de una tarjeta en el sistema.

El saber leer el contenido y organización de los diferentes tipos de ficheros de sonido existentes no se explica por quedar fuera de tema , pero proximamente , van a aparecer artículos destinados a este tema , por lo que los usuarios preocupados en esta materia , pueden respirar tranquilos.

CORREO DEL LECTOR

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relaccionado con la programación.

P *Hola, quiero empezar a programar en ensamblador y en C++ y tengo las siguientes dudas:*
- Para programar en ensamblador, ¿que requerimientos de hardware y software precisaría?
- Sin conocimientos previos de este lenguaje, ¿tendría muchas dificultades de comprender los artículos aparecidos en su revista, desde el primer número?
- Los compiladores de C++ habituales, ¿son compatibles entre ellos?, ¿utilizan las mismas normas sobre C++?
Un saludo y gracias anticipadas.

Juana Dolores Ortega
(Petrel / Alicante)

R El ensamblador, al tratarse del lenguaje "base", lógicamente requiere menos recursos para funcionar o para compilarse (ensamblarse en este caso) que cualquier otro lenguaje de programación. De esta manera, con un simple 8086 se podrían ensamblar fuentes muy extensos y en un tiempo mínimo, pues el ensamblador sólo tiene que ir convirtiendo cada instrucción escrita en su correspondiente opcode sin tener que interpretar, optimizar o reordenar las distintas sentencias. De cara al software necesario, el único requisito es el ensamblador, que viene acompañado de su depurador y de su enlazador correspondiente (estos últimos suelen ser los mismos que los incluidos en los paquetes de compiladores C).

Los artículos publicados en la revista son para eso, para comprenderlos por el mayor número posible de lectores. En su caso, lo más apropiado es empezar con el curso de ensamblador en vigor, consiga los números que le falten y vaya secuencialmente, no debería tener problemas.

Desde hace un tiempo, realmente no mucho, los compiladores de C++ siguen las normas dictadas por AT&T, quienes definen (conjuntamente con todos los grandes desarrolladores) las nuevas funciones a implementar. Actualmente todos los compiladores C++ siguen el estándar AT&T 3.0, por lo que cualquier fuente escrito y compilado bajo uno de ellos, puede migrarse a otro sin mayores problemas. Para ello no deben utilizarse funciones ni librerías exclusivas del compilador, como es el caso clásico de las funciones gráficas. Por suerte en la descripción de cada función, siempre viene indicado a que norma se adapta (ANSI C, C++ 3.0, etc.)

P *Estaría agradecido si me facilitaran información sobre la interrupción que controla al ratón, con el fin de poder implantarlo en mis programas. También me serían muy útiles las relativas al control del altavoz del PC, y datos sobre el formato de los archivos de sonido del tipo VOC.*

José B. Castro
(Cangas / Asturias)

R La interrupción "clásica" de manejo del ratón es la 33h, pero no está implementada en el sistema operativo (DOS) y es necesario un controlador o driver para tener estos servicios disponibles. Las principales funciones a las que se puede acceder mediante dicha interrupción son:

0 Reinicializa el controlador del ratón

1 Activa el ratón en pantalla

2 Desactiva el ratón en pantalla

3 Informa de la posición del ratón y del estado de sus botones

4 Fija una posición para el ratón

5 Informa del estado de las pulsaciones en los botones, pulsaciones realizadas y coordenadas de la última pulsación

6 Informa del estado de las "sueltas" o liberaciones en los botones, "sueltas" realizadas y coordenadas de la última liberación

7 Define un rango horizontal para el ratón

8 Define un rango vertical para el ratón

9 Define la forma del cursor en modo gráfico

10 Define la forma del cursor en modo texto

15 Define la sensibilidad del ratón

22 Salva el estado del controlador del ratón

23 Restaura el estado del controlador del ratón

La llamada a cualquiera de estas funciones es muy simple y sólo se deben cargar los registros con los valores correctos y generar la interrupción. Un ejemplo en Pascal sería:

```
procedure LeePosRaton(var x, y : word; var boton1, boton2 : boolean);

var regs : registers;
begin
   regs.ax := 3;
   Intr($33, regs);
   x := regs.cx;
   y := regs.dx;
   boton1 := (regs.bx and 1) <> 0;
   boton2 := (regs.bx and 2) <> 0;
   end;
```

Toda esta información no es muy útil si se desea trabajar en modos de SVGA, pues la mayoría de los drivers no contemplan estos modos y se hace necesario el programarse un propio conjunto de rutinas de manejo del ratón.

El altavoz del PC sólo tiene dos estados: encendido o apagado y, o bien se le manipula directamente, o se puede programar al timer 2 para que él vaya manejando al altavoz según su programación.

El control del altavoz se realiza por medio del puerto 61h, mediante sus bits 0 y 1.

Con el bit 0 se controla su manera de operar:

- 0 El estado del altavoz se controla con el bit 1 (bit 1 a 1 = encendido, bit 1 a 1 = apagado)

- 1 El altavoz se conecta al canal 2 del timer PIT y el bit 1 se utiliza como controlador de la conexión (0 = no conectado, 1 = conectado).

Para generar un simple tono sólo hay que hacer oscilar la salida del altavoz a una determinada frecuencia (encenderlo y apagarlo), a simples rasgos quedaría de la siguiente manera:

```
LEER valor del puerto 61h
DO
{
ESCRIBIR en el puerto 61h el valor antiguo OR 2
RETARDO
ESCRIBIR en el puerto 61h el valor antiguo
RETARDO
}
WHILE TeclaNoPulsada
```

Claramente esta no es una manera eficiente de realizar sonidos y se debe utilizar el timer 2 para conseguir efectos mejores.

Acerca del formato de los ficheros VOC, se está preparando una completa serie de artículos referentes al tema, que aparecerán publicados en breve.

P *Me pongo en contacto con ustedes, ya que llevo algún tiempo pensando como puedo conectar mi ordenador con el mundo exterior, por ejemplo: como puedo accionar un motor desde el PC ¿es esto factible? Si lo fuera, ¿podrían indicarme como? Aprovecho para felicitarles por el gran trabajo que están realizando.*

Luis Alvarez Romero
(Zaragoza)

R Las posibilidades de control de automatismos desde el PC es realmente enorme. Teniendo en cuenta que, por ejemplo, en el puerto paralelo se dispone de más de ocho señales modificables (cada pin del conector se puede poner a 5 voltios o a 0), y que esto se consigue con sólo una simple escritura en un puerto (o una llamada a la función correspondiente); es evidente que con la conexión de algún pequeño relé (mucho cuidado con el consumo de estos aparatos, pues pueden exigir demasiada intensidad al puerto y quemarlo) se pueden conseguir verdaderas maravillas. Si se quiere ir más allá, se debe recurrir a placas especiales con DACs (Digital to Analog Converter) en su interior, con lo que se controlan tensiones de salida variables, posibilitando cualquier control preciso sobre equipos eléctricos. Se están recibiendo bastantes peticiones sobre el tema y en breve intentaremos publicar un artículo sobre automatismos desde el ordenador. De todas maneras, este tema es muy delicado y cualquier prueba puede saldarse con averías en el PC, por lo que sólo deberían probarlo personas muy acostumbradas al manejo de aparatos electrónicos y hardware.

P *Llevo mucho tiempo oyendo hablar de programación y arquitecturas Cliente/Servidor, más o menos dispongo de una idea sobre su filosofía, pero me pierdo cuando intento comprender requisitos técnicos. ¿Que tipos de lenguajes se utilizan para programar?, ¿donde se encuentran los ejecutables? Supongo que habrá más lectores interesados en el tema. Gracias anticipadas.*

Belén Hebra
(Alcorcón / Madrid)

R Este modo de funcionamiento o arquitectura tiene infinidad de variantes, pero básicamente consiste en programas "Clientes" que se encargan de gestionar la conexión con el usuario, ejecutados generalmente en PCs y que realizan peticiones de datos a programas "Servidores" que se encuentran en ordenadores remotos o Host. Los Clientes suelen programarse en C o lenguajes "visuales" tan de moda ahora y los Servidores en Cobol, RPG o cualquier lenguaje clásico de los grandes sistemas. También se aplica esta arquitectura en redes de PCs, con uno o varios de ellos "sirviendo" al resto. En estos casos también se realizan en C las peticiones a las bases de datos residentes en el Servidor.

No hay que confundir a los servidores de ficheros, como el servidor de una red local; con el servidor de un sistema Cliente/Servidor, ya que este último realiza transacciones y lanza programas a petición de "sus" clientes.